FACE DETECTION SYSTEM

Prepared by
CHAN BEH TENG
WEK 010028

Under the Supervision of
Pn. Azwina Mohd. Yusof

Under the moderation of
Pn. Siti Hafizah Ab. Hamid

Abstract

The aim of this project is to replicate on a computer what human beings are able to do so effortlessly - detect the presence of faces in their field of vision. To a layman, it may appear trivial, but to actually implement the necessary steps leading to a successful execution of this in an algorithm is difficult and still an unsolved problem in computer vision.

The proposed Face Detection System is an application that automates the detection of frontal human face(s) in a given image. The system performs all tasks including image pre-processing, analysis, and output. The user merely presents it with the image. The system takes as input a 24 bit RGB colour image which contains face(s). Before the detection process begins, the system pre-processes the image into a format required by the system. The system will generate an output where every detected face will be indicated with a rectangle on a copy of the original image.

The method used for detecting face is a combination of colour-based technique and template matching. A gray-scale image is generated from the original input image. Skin regions are then separated from non-skin regions by segmenting the gray-scale image. After that, the system will locate the frontal human face(s) within the skin regions, using template matching.

The Face Detection System is implemented in Visual Basic 6.0. The system was developed based on the Unified Process, which uses object-oriented paradigm as the basis of software development.

Acknowledgement

Taking up this project has been a very challenging experience. I would not have pull through without the help of numerous people.

First and foremost, I thank my supervisor Puan Azwina Mohd.Yusof for her guidance and advice. Her words of encouragement has given me the confidence to take up such an interesting yet challenging system. She has been very understanding and patience in answering all my doubts and questions. Special appreciation to Mr Chiew Thiam Kian for moderating the report and VIVA I presentation. I would also like express my gratitude to Pn. Siti Hafizah Ab. Hamid for moderating the WXES 3182 report and the VIVA II presentation.

Special thanks to ex-seniors who provided me with helpful tips. My heartfelt appreciation to my dearest course mates and friends who offered me a shoulder to cry on when things get too tough.

And to Kendrick, thank you for all the love and support.

Thanks to all of you!

Chan Beh Teng
February 2004

TABLE OF CONTENT

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Project Overview

The current evolution of computer technologies is steering towards applying artificial intelligence in machines in order for the machine to duplicate human cognitive abilities. Computer vision, for example aims to duplicate human vision. One particular discipline in computer vision which is garnering tremendous amount of research is face detection. From the many researches done, a number of methods have been proposed to detect face in images.

Over the past ten years face detection has been thoroughly studied mainly due to the reason that it has a number of interesting applications in various fields such as security, earth sciences, and image communication. For instance face detection is a crucial step in face recognition which can be used to restrict the access to high-security areas. Video surveillance systems could also use this technique to detect the face of delinquent caught in the act, which could reduce the time spent in identifying and tracking criminals. Another application is the use of such technique to process satellite images to detect a particular pattern. Not only national armies are very interested in this application but also scientist who can make use of that tool to study the surface of our planet. Videoconference systems can also benefit from face detection for their coding schemes. The list of possible applications is very long, but

the few examples mentioned above already demonstrate the significance of face detection.

No matter how simple and trivial it may sound, face detection in images is a challenging task. In pattern recognition parlance, human face is a complex pattern. Different poses and gestures of the face accentuate complexity. Variability in scale, location, occlusion, and lighting conditions also change the overall appearance of faces. In fact, Most in the past research work on face detection focused on detecting frontal faces thus leaving out the problem of pose invariance. Although there is still some space for improvement on frontal face detection, the key issue of current and future research is bound to be pose invariance.

1.2 Project Motivation

As face detection, or any subjects related to face processing, is not offered as a study course in the faculty, this project serves as a good opportunity to study face detection and its existing methodology in depth. In the end, the theoretical knowledge is applied to a practical situation. In this case, it will be to implement a face detection system using the existing methodology. Therefore, this project is taken up purely for educational purposes.

The system is built to detect the presence of human face(s) in a 2-dimensional static image. The system will take as input colour RGB 2-D images in bitmap format. The system will process the input image accordingly and after that the algorithm implemented in the system will be executed to detect human face(s) in the image.

The detected human face(s) will be indicated by a marking (a red spot, a rectangle, etc.)

1.3 Objective of the Project

- To study and review the existing methodology for face detection, and how to implement algorithms to perform the process of face detection.

- To choose one or a combination of existing techniques/methods in building a realistic software program for the purpose of face detection.

- To create, in general, a useful software program to perform the task of face detection with a good success rate.

1.4 Scope of the Project

The scope of the project includes designing and building a system that is able to detect the presence of human face(s) in a given image and indicate / mark the detected face(s) as output. An algorithm which is made up of a combination of skin colour segmentation and template matching shall be implemented in order to perform face detection. The system takes as input 2-dimensional 24 bit colour RGB images in bitmap format. The face(s) in the input image has to be frontal view of the image and the number of faces in the image is approximately between 1 to 20. The height of the image must not exceed 239 pixel unit and the width must not be more than 319 pixel unit.

This project is initiated exclusively for educational purposes and not to build a commercial system/software. Therefore, there are **limitations** in this project compare to other existing commercial system. One being the execution time of the system is not taken into account when measuring its performance. The system is meant for detecting faces in static images only and does not support dynamic images like video.

For the Face Detection System, it is implemented based on existing techniques and algorithm and to date researchers has yet to discover a technique that will yield perfect results. Therefore, it is near impossible for the Face Detection System to generate 100% accuracy in detecting face.

Moreover, as the terminology for various face processing task (face detection, face localization, face recognition, etc.) are often misunderstood, it is be stressed that the system only detects the presence of face and does not extend to recognizing/ identifying the face

1.5 Expected Outcome

This project, when completed, will be a software program implemented using one or a combination of the existing methodology for face detection. An algorithm will be implemented in the software program which will scan a digital image of a colour photograph containing human face(s) to detect the presence of human face(s). A rectangle will be drawn on around every detected face in the original colour image.

1.6 Project Schedule

This Face Detection System project took 2 semesters to be completed. The following Table 1.1 and Figure 1.1 details the activities that were throughout the 2 semesters.

Table 1.1: Face Detection System project schedule

	Task Name	Duration	Start	Finish	Predecessors
1	Literature Review	30 days	Tue 03-05-10	Wed 03-07-09	
2	Methodology Research	15 days	Wed 03-06-25	Wed 03-07-09	
3	Inception	15 days	Thu 03-07-10	Thu 03-07-24	2
4	Life-Cycle Objectives Mile	0 days	Tue 03-06-10	Tue 03-06-10	
5	Elaboration(first iteration)	18 days	Fri 03-07-25	Mon 03-08-11	3
6	Elaboration(second iteratio	15 days	Tue 03-08-12	Tue 03-08-26	5
7	Lifecycle Architecture Mile	0 days	Wed 03-09-03	Wed 03-09-03	
8	Construction(first iteration)	78 days	Wed 03-08-27	Fri 03-11-14	6
9	Consuction(second iteratic	67 days	Sat 03-11-15	Wed 04-01-28	8
10	Initial Operation Capability I	0 days	Tue 03-06-10	Tue 03-06-10	
11	Transition	10 days	Thu 04-01-29	Sat 04-02-07	9
12	Product Release Milestone	0 days	Tue 03-06-10	Tue 03-06-10	
13	Documentation	234 days	Tue 03-06-10	Sun 04-02-08	

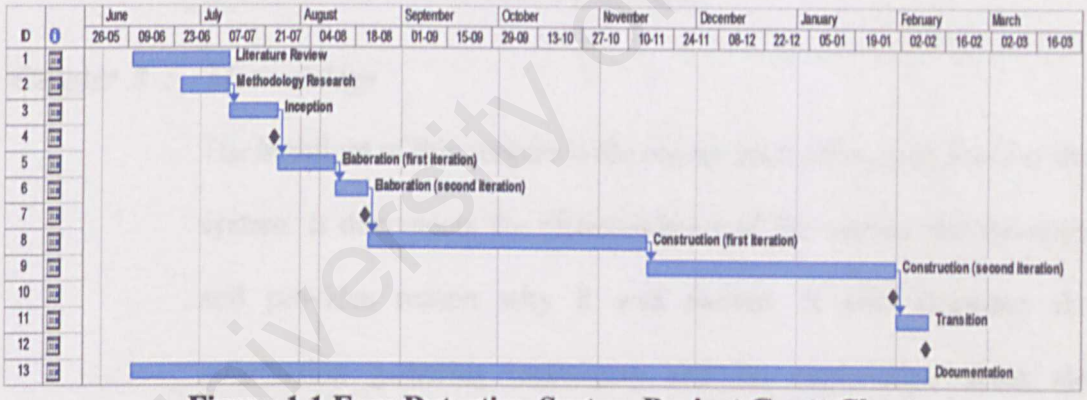


Figure 1.1 Face Detection System Project Gantt Chart

1.7 Report Layout

This project report is made up of 8 chapters. The following text is a layout of the organization of the whole report. It gives an overview of the major issues involved for the planning and implementation of the project

Chapter 1 : Introduction

This chapter gives an overview of the proposed system for the project. It defines the proposed system and presents vital information regarding the project which includes the project objective, project scope, project motivation and project schedule.

Chapter 2 : Literature Review

This chapter is the most information-packed section of this report. All research and studies that was conducted are documented in this chapter. It provides a summary on issues and topics related to the proposed system. It is the combination between literature search and literature review. Among the topics discussed are development tools and technology including operating system, programming languages and as well as face detection techniques

Chapter 3 : Methodology

The highlight of this chapter is the chosen methodology to develop the system. It documents the characteristics of the chosen methodology and provides reason why it was chosen. It also discusses the information gathering techniques and the explanation about the development software and platform chosen to develop this system.

Chapter 4 : System Analysis

This chapter describes the system analysis of the project including techniques used to capture requirements, specification of the functional and non-functional requirements, as well as hardware and software requirements.

Chapter 5 : System Design

This chapter provides an overview of the overall architecture of the proposed system. It explains the conceptual and technical design of the system. Diagrams are used to model all aspects of the system.

Chapter 6 : System Implementation

This chapter explains in detail how the algorithm for skin segmentation and template matching, which are methods used in the Face Detection System to detect face, is implemented and coded.

Chapter 7 : Testing

This chapter shows explains what testing approach was used to test the system. The test set data are provided in table form. A discussion section was included to explain some bizarre test result and error encountered.

Chapter 8 : System Evaluation and Conclusion

This chapter evaluates the system in terms of its performance as tested and also included suggestion to apply the system for future use.

1.8 Chapter Summary

This chapter defines the project and the proposed system. It started off with a brief introduction to topic of face detection. Apart from that, important issues regarding the project were outlined. They were discussed in subtopics which include Project Objectives, Project Motivation, Project Scope, Expected Outcome, Project Schedule, and Report Layout. In the following chapter, a literature review will be carried out to document all information regarding the proposed system.

CHAPTER 2

Literature Review

2.1 Introduction

The purpose of conducting this literature review is to provide a thorough understanding on the domain of the Face Detection project by looking at publications by accredited scholars and researchers. It can be perceived as the pre-analysis phase of the project where different methods of face detection are studied and compared in order to determine the most feasible method to be adopted for this Face Detection project. Research has also been done on existing system related to face detection so as to have a clear view of the application of face detection in the real world. Therefore, many requirements of the Face Detection system can be captured from the literature review.

2.2 Domain Studies

Images containing faces are essential to intelligent vision-based human computer interaction. Hence, many research has been conducted on face processing which includes face recognition, face tracking, pose estimation and expression recognition. However, the first step of every face processing system is detecting the locations of

images where face is present. Below are the findings of all aspects related to face detection.

2.2.1 Definition

Face is defined as front of the head from forehead to chin. For the purpose of this project, *face* refers to only the human face.

Face analysis can be categorized into different areas such as detection, recognition, tracking, expression recognition and modelling.

Face detection is to determine whether or not there are any faces in a given arbitary image and, if present, return the image location and extent of each face [1]. Face detection and ***face tracking*** are related subjects in stand alone application such as counting faces and surveillance, where the location and orientation of face are continuously estimated.

Face localization aims to determine the image position of a single face. This is a simplified face detection with the assumption that an input image only contains one face

Face recognition deals with identifying the identity of a person in an image or a sequence. A typical application for face recognition is access control where the face of a user is matched against a pre-stored image of the same user instead of a username or password. Face recognition has numerous potential application: criminal identification, workstation security, video-mail retrieval and building security.

Facial expression recognition concerns identifying the affective states (happy, sad, disgusted, etc) of humans [2]. Therefore, its goal is to detect the presence and location of features such as eyes, mouth, nose, nostrils, ears, etc.

2.2.2 Challenges Associated With Face Detection

Given an image, the goal of face detection is to identify all image regions that contain a face regardless of its orientation, three dimensional position, or lighting condition. However, there are factors that makes it difficult to achieve this goal. The challenges associated with face detection can be attributed to the following factor [2]:

- **Pose.** The images of a face vary due to the relative camera-face position (45 degrees, frontal, etc.) and some basic facial features (mouth, nose, etc.) may be partially or wholly occluded.
- **Presence and absence of structural component.** Facial features such as skin tone, beard, mole, glasses may or may not be present and these components have a wide variety including shapes, colour and sizes.
- **Facial expression**. The appearance of faces is directly affected by the person's facial expression (frowning, smiling, sulking, etc.)
- **Occlusion.** Faces may be partially occluded (hidden) by certain objects. In an image with a group of people, a face may be partially occluded by another face, causing overlapping face images.

- **Imaging conditions.** When the image is formed, factors such as lighting and camera characteristics (lenses, sensor response) may affect the appearance of a face.

2.2.3 Methods In Face Detection

The idea of using computation to "detect" face is very intriguing. This idea has led to many interesting algorithms and techniques for face detection, each with their own uniqueness. In this section, existing techniques to detect faces is reviewed. As stated in Chapter 1, the scope of this project includes detecting faces in still images and does not include moving images like video, hence the methods discussed are for static image only. According to Yang and Kriegman [1], the methods can be classified into four major categories:

1. **Knowledge-based method.**

   These are rule-based methods whereby human knowledge of what constitutes a face is encoded. The rules usually capture the relationship between features.

2. **Feature-invariant approaches.**

   Designed mainly for face localization, these methods aim to search for structural features of a face image that exist, regardless of the variantion in pose, viewpoint, or lighting conditions. These information is then used to locate faces.

3. **Template-matching methods.**

Several standard patterns of a face are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection.

4. **Appearance-based methods.**

In contrast to template matching, these methods have models (or templates). The models are learned from a set of training images which should capture the representative variabilities of facial appearance. These learned models are then used for face detection.

Table 2.1 summarizes the algorithms and representative works for face detection in a single image for these four categories.

Table 2.1 : Categorization of Methods is Face Detection For Single Image

Approach	Representative works
Knowledge-based	Multiresolution rule-based method
Feature invariant	
- Facial Features	Grouping of edges
- Skin Colour	Mixture of Gaussian
- Multiple Features	Integration of skin colour, size and shape
- Texture	Space Gray-Level Dependence matrix(SGLD) of face pattern
Template matching	
- Predefined face template	Shape Template
- Deformable templates	Active Shape Model (ASM)
Appearance-based method	
- Eigenface	Eigenvector decomposition and clustering
- Neural Network	Ensemble of neural networks and arbitration schemes
- Support Vector Machine (SVM)	SVM with polynomial kernel
- Hidden Markov Model (HMM)	Higher order statistics with HMM
- Information-Theorethical Approach	Kullback relative information

The feature invariant method and the appearance-based method are two of the categories which are more feasible to be used for this project as they are more popular, hence abundant references are available. Below are the review of some of the more general approach to these two methods.

2.2.3.1 Face Detection Using Neural Network

Neural networks have been applied successfully in many pattern recognition problems, such as object recognition and optical character recognition. Face detection can be treated as a two-class pattern recognition problem (face image or no-face image), hence various neural network architecture has been proposed.

Artificial neural network has been developed as generalization of the mathematical models of human neural biology. A neural network consists of a large number of simple processing elements called neurons. Each neuron is connected to other neurons by beams of directed communication links, each with an associative weight. The weight represents information being used by the net to solve problem. The method of setting the values of weights (training) is an important characteristic of different neural nets. There are two types of training - supervised and unsupervised.

Among all the face detection methods that used neural nets, the most significant work is done by Rowley, Baluja and Kanade [3]. It works in the following way : first, the image is scanned at different scales and a set of neural-network-based filters is applied to each block of pixels. Once this is done, it will generate a list of possible face locations in the images but some of them correspond to the same face or are false positives (false identification of a face). This problem is alleviated by a bootstrap method that selectively adds image to a training set as training progress. Starting with a small set of non-face examples in the training set, a multilayer perceptron network classifier is trained with this database of examples. Then, the face detector is run on a sequence of random images and all the non-face patterns

that the system wrongly classifies as face are collected. These false positives are then added to the training database as new non-face examples. This bootstrap method avoids the problem of explicitly collecting a representative sample of non-face patterns and has been used in later works.

The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. However, these methods have two major drawbacks for this project. Firstly, the neural network should be trained with at least a few thousands of face and non-face images which will take months to be built. Such database already exists and is available on the internet but it is difficult to find one that matches this project's needs.

2.2.3.2 Face Detection Using Support Vector Machine

Support Vector Machines(SVM) is a learning technique developed by V. Vapnik and his team (AT&T Bell Labs) that can be seen as a new method for training polynomial, neural network, or radial basis functions classifiers. The decision surfaces are found by solving a linearly constrained quadratic programming problem. This problem is challenging when the size of the data set becomes larger than a few thousands. A large scale problem of the type posed by SVM can be solved by a decomposition algorithm.

While most methods for training a classifier are based on the minimizing the training error, SVMs operate on another induction principle called *structural risk*

minimization [1], which aims to minimize an upper-bound on the expected generalization error.

From the simple case of two linearly separable classes, a good choice is the hyperplane that leaves the maximum margin between the two classes. The margin is defined as the sum of the distances of the hyperplane from the closest point of the two classes. It turns out only a small number of data points are relevant to the hyperplane. All the other data points could be deleted from the data set. These small number points are called support vectors. Intuitively, the support vectors are the data points that lie at the border between the two classes. The number of support vectors is usually very small.

In real problem, we cannot solve all of them with a linear classifier, so the techniques have to be extended to allow for non-linear decision surfaces. Projecting the original set of variables in a higher dimensional feature space easily does this.

SVMs are very well-founded from the mathematical point of view, being an approximate implementation of the structural risk minimization induction principle.

2.2.3.3 Eigenfaces

Human face images are very similar in all configurations and they can be described by some 'basic face images'. Elaborating on this idea, one can find the 'basic faces' that best account for distribution of face images within the entire face

space using the principle component analysis (PCA). The basic faces are better-known as *Eigenfaces*.

Turk and Pentland [8] applied PCA to face recognition and detection. Principle component analysis (also known as the Karhunen-Loeve transform) is performed on a training set of face images to generate the eigenvectors. These eigenvectors can be thought of as a set of features that together characterize the variation between face images. When the eigenvectors are displayed, they look like a ghostly face, and are termed eigenfaces. The eigenfaces can be linearly combined to reconstruct any image in the training set exactly. In addition, we can find the average of the eigenfaces, which can be used to compare with test images to locate the face in an image. Figure 2.1 is the first 20 eigenfaces of 128 test images that was gathered [9].

Figure 2.1: Standard Eigenfaces

The eigenfaces span a subspace (called the face space of the image space). Images of faces are projected onto the subspace and clustered. Similarly, non-face training images are projected onto the same subspace and clustered. Images of faces do not change radically when projected onto the face space, while the projection of non-face image appear quite different. To detect the presence of a face in a scene, the distance between an image region and the face space is computed for all locations in the image. The distance from face space is used as a measure of 'faceness', and the result of calculating the distance from face space is a 'face map'. A face can then be detected from local minima of the face map. One of its advantage is that building eigenfaces does not require big training set.

However, this method only works well for face detection if there is only one face in the image. Therefore, this approach is usually adopted for face recognition system where it is able to search for a face in the database which is similar to the input image, without the need of a preprocessor to detect the presence of face.

As the eigenface method is moderate in complexity yet powerful, it is usually combined with other existing face detection methods for detecting multiple faces in an image.

2.2.3.4 Face Detection Using Skin Colour

Human skin colour has been used and proven to be an effective feature in many applications from face detection to hand tracking. It was found that different human skin colour gives rise to a tight cluster in colour spaces even when faces of different

races are considered. This means color composition of human skin differs little across individuals.

Several colour spaces have been utilized to label pixels as skin including RGB, HSV, YES and YIQ. The famous method for face detection using skin colour is to utilize HSV colour space. Figure 2.2 shows the colour distribution of skin pixels which is plotted by Sherrah and Gong [7], the human skin colour shows a distinct distribution in HSV colour space. It is possible to segment skin regions by properly thresholding the Hue/Saturation value of each pixel.

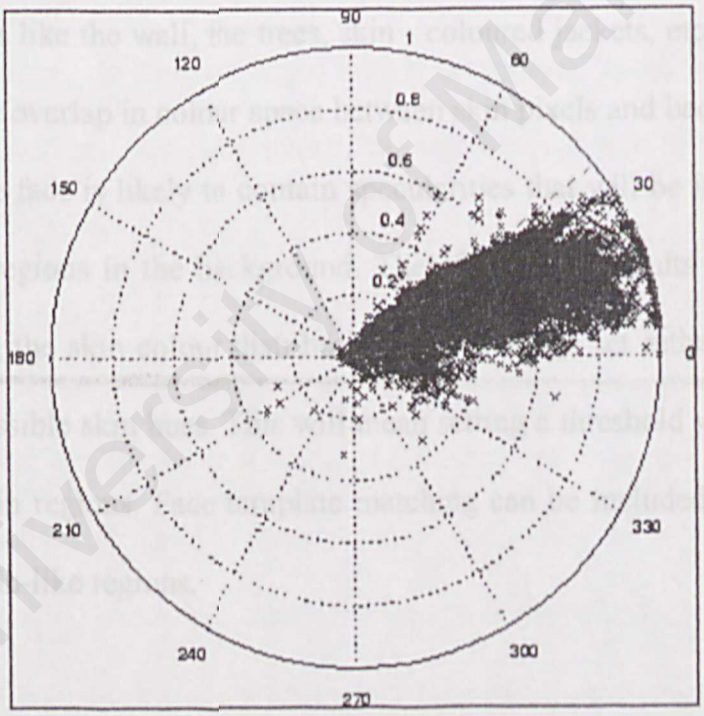


Figure 2.2 : Skin pixels plotted in HS-space

Saxe and Foulds [6] proposed a method that uses histogram intersection in HSV colour space. An initial patch of skin colour pixels called the control seed, is chosen by the user and is used to initiate the iterative algorithm. To detect skin colour regions, this method moves through the image, one patch at a time and presents the control and current histogram from the image for comparison. If the number of instances in common is greater than a threshold, the current patch is classified as being skin colour.

The method of face detection using skin colour does well in marking out the areas, where there is actually skin (i.e. faces and hands). But it also mark the point of unwanted objects like the wall, the trees, skin - coloured jackets, etc. It is inevitable that there will be overlap in colour space between skin pixels and background pixels. For example, the face is likely to contain specularities that will be indistinguishable from white-ish regions in the background. Therefore better results are obtained by tightly modeling the skin colour distribution in *this* image set rather than trying to cope with all possible skin hues. This will mean setting a threshold value to segment skin and non-skin regions. Face template matching can be included at this point to locate face in skin-like regions.

This method is more applicable because it is able to detect multiple faces in an image. However , it is found that skin colour alone will not be sufficiently reliable to specifically identify human faces in a scene likely to contain skin-look-alike background. Therefore this method ought to be combined with other sources of information such as shape and appearance in order to be truly effective.

2.2.3.5 Template matching

Low-level analysis (skin colour, edge)-based, motion, etc. are likely to generate ambiguous feature which lead to inaccuracy in the detection of face. This problem is usually solved using a higher -level feature analysis method called template matching. Template matching employs the knowledge of face geometry to characterize and subsequently verify various features from their ambiguous state.

In template matching, a standard face pattern (usually frontal) is manually predefined or parameterized by function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently. The existence of a face is determined based on the correlation values. This method can be further elaborated based on the type of template used to model the facial features.

The simpler and one of the earlier approach will be using a pre-defined template where the template is predefined in terms of line segment. The correlations between templates are computed first to detect candidate locations of faces. Then, the candidate position is further analyzed to reject or accept the face candidate. The second approach is to use active shape models. One of the more popular active shape model will be the deformable templates, introduced by A.Yuille. In this approach, facial features are described by parameterized templates. Yuille incorporated global information of the eye to improve the reliability of the feature extraction process. A deformable eye template based on its salient features is parameterized using 11 parameters. The template once initialized near an eye feature will deform itself toward optimal feature boundaries. An energy function is defined to link edges,

peaks, and valleys in the input image to corresponding parameters in the template. The best fit of the elastic model is found by minimizing an energy function of the parameters.

The template matching approach has the advantage of being simple to implement. However, using this method alone for face detection system is inadequate since it cannot effectively deal with various scale, pose and shape. Therefore it is usually incorporated into other face detection algorithm and this has been proven to yield excellent results.

2.2.4 Existing System Review

There are numerous face detection system available in the market to date. They fall into two categories; commercial system and coursework project system. The commercial systems are built by renowned companies specializing in biometrics applications, whereas the coursework project system are built by students and researchers in universities. The latter ones are built for educational and research purposes, so detailed information about the structure of the system are readily available compared to the commercial systems.

5 different systems have been reviewed. All the systems are either solely for detecting face or has combination of multiple face processing (face identification, face localization, etc.).

The objective of this review is to understand the common features and functionalites offered by existing face detection system. Therefore it is one way to capture requirements for this project. However, it is of utmost importance that this review provides an insight on how different face detection method are applied in the algorithms for building a face detection system and also the corresponding results of the system.

FaceIt

FaceIt is a facial recognition software which was built by Visionics [14]. The $30,000 system was loaned to Tampa Police Department in July 2001 to increase surveillance of city residents and tourists. The FaceIt software was deployed via 36 security cameras stationed at strategic locations in Tampa district. FaceIt allows snapshots of faces from the crowd to be compared to a database of criminal mugshots.

Visionics, a company based in New Jersey, is one of many developers of facial recognition technology. In year 2002, Visionics merged with Identix, opting for the Identix name. **FaceIt** can detect and locate a person's face out of a crowd, extract that face from the rest of the scene and compare it to a database full of stored images.

FaceIt defines every distinguishable landmark of human face as nodal points (distance between eyes, width of nose, cheekbones,etc.). These nodal points are measured to create a numerical code that represents the face in a database. This code

is called a **faceprint**. Only 14 to 22 nodal points are needed for the FaceIt software to complete the recognition process.

Below are the basic process that is used by FaceIt system to detect, capture and compare face images.

1. **Detection** - When the system is attached to a video surveillance system, the recognition software searches the field of view of a video camera for faces. If there is a face in the view, it is detected within a fraction of a second. A **multi-scale algorithm** is used to search for faces in low resolution. (An algorithm is a program that provides a set of instructions to accomplish a specific task). The system switches to a high-resolution search only after a head-like shape is detected.

2. **Alignment** - Once a face is detected, the system determines the head's position, size and pose. A face needs to be turned at least **35 degrees** toward the camera for the system to register it.

3. **Normalization** -The image of the head is scaled and rotated so that it can be registered and mapped into an appropriate size and pose. Normalization is performed regardless of the head's location and distance from the camera. Light does not impact the normalization process.

4. **Representation** - The system translates the facial data into a unique code. This coding process allows for easier comparison of the newly acquired facial data to stored facial data.

5. **Matching** - The newly acquired facial data is compared to the stored data and (ideally) linked to at least one stored facial representation.

The heart of the FaceIt facial recognition system is the **Local Feature Analysis** (LFA) algorithm. This is the mathematical technique the system uses to encode faces. The system maps the face and creates a **faceprint**, a unique numerical code for that face. Once the system has stored a faceprint, it can compare it to the thousands or millions of faceprints stored in a database. Each faceprint is stored as an **84-byte file.**

Figures 2.3 and 2.4 are excerpts from the FaceIt system. It depicts the system identifying an assailant whose crime was recorded by the surveillance camera. His face was matched with the profile of a serial criminal.

Figure 2.3: Face detected and database is searched for match in FaceIt

Figure 2.4: Match found by FaceIt

While the software proved reliable in testing, the Tampa police said they were unable to make any positive identifications or arrests as a result of its use. The American Civil Liberties Union vigorously opposed its deployment from the outset, arguing that the system was prone to false matches and that it had never positively identified any missing or wanted people whose images were in the department's databases. The increased surveillance on residents and tourists had also creates objections from privacy rights groups.

Therefore the Tampa Police Department announced in August 2003 that they will not renew its facial recognition software contract with Identix Inc [10].The FaceIt system was ultimately phased out from the department.

Photobook Demo

Most face recognition programs or experiments conducted test on at most just a few hundred faces. The Photobook demo [9] deals with a larger database of face images belonging to approximately 3000 people.

This program was built using principal component analysis to approximate the eigenfaces for the database. The database can be interactively searched using an X-Window browsing tool call Photobook. The user can select different types of face with specifications on features; *e.g.*, senior Caucasian males with mustaches, or adult Hispanic females with hats. An object-oriented database is used to search through face images to find subsets that matched user's specifications. Top matches found in the database are presented to user where user will select the most similar face. The remainder of the database images can be viewed by ``paging" through the set of images. Photobook will then use the eigenvector description of that face to sort the entire set of faces in order to find the most similar face image to that of the chosen one. Photobook then re-presents the user with the face images, now sorted by similarity to the selected face. The face images found are then presented for viewing. The images are sorted by similarity to the selected face where similarity decreases left to right, top to bottom.

The Photobook can also be extended to support modular eigenspaces such as eigen eyes, eigen mouths, and eigen noses.

The Figure 2.5 below shows the typical search results of Photobook. The face at the upper left of each set of images was selected by the user; the remainder of the

faces are the 15 most-similar faces from the database, which belongs to the same person. The most important feature of Photobook is its ability to find the same person despite wide variations in expression and variations such as presence of eye-glass.

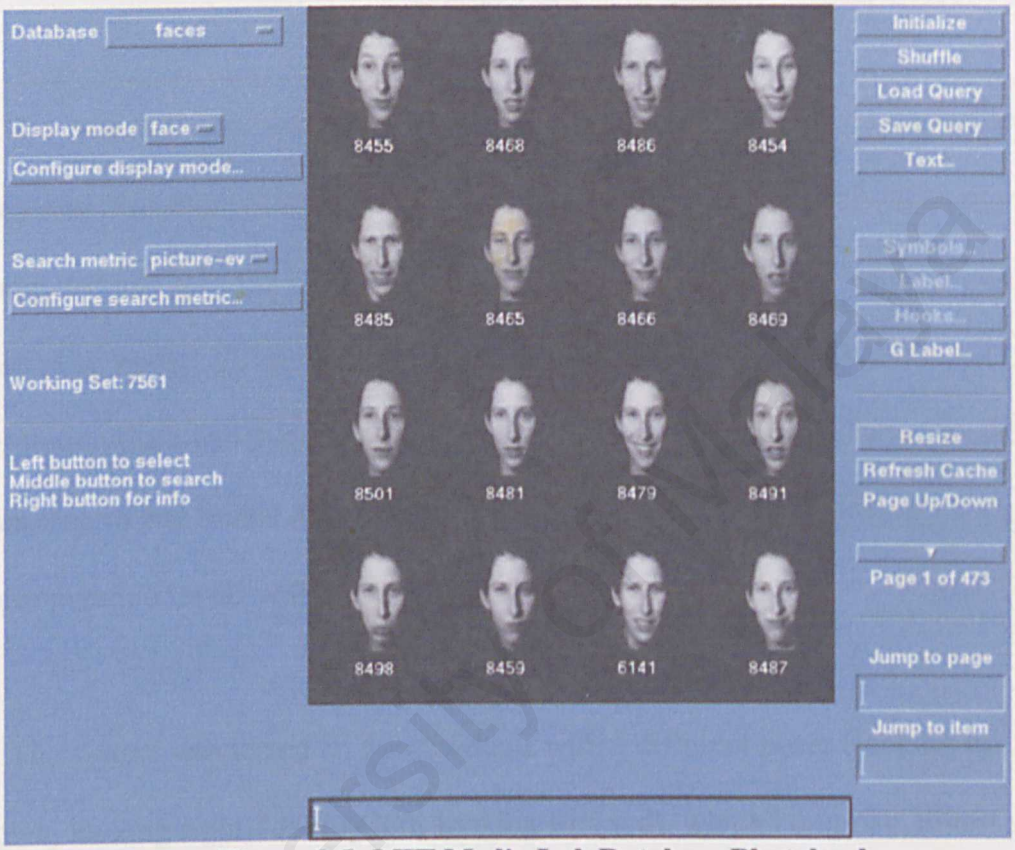


Figure 2.5: MIT Media Lab Database Photobook

Face Recognition Using Eigenfaces and Gaussian Pyramid [12]

This is a simple face recognition system built by Christopher James Cobb (2001). The method which was used to implement the system is the PCA (Principle Component Analysis) method which involves the application of eigenfaces. Although the system was called "Face Recognition", what the system does is to extract relevant information from the face, encode it, and compare one face encoding

with a database of similarly encoded images in order to detect the face. It does not identify the identity of the person. The image used in the system is upright, forward-looking face in an image, where the face could be presented at different scales. The program was implemented in Matlab which returns the image with a box around the face.

The first task was to detect face using the eigenfaces method of Moghaddam and Pentland. The eigenvectors of the covariance matrix of the training set of images are computed and then the eigenfaces are used to compare with test images to locate the face in an image. To handle the different scales at which a face might appear, a multi resolution Gaussian pyramid was formed from the input image. The original image was blurred and scaled down by a factor of two. This was done multiple times to form pyramid levels with scales an octave apart.

The system was tested on 4 individuals with 5 different poses (looking up, looking down, looking right, looking left, looking forward), which comprise 20 test samples. The program accurately assesses the face as it is facing forward, but when the head is tilted to the left or the right the results will vary.

In figure 2.6 which was one of the system's test image, when the person's head is tilted to the left and right, one of the eyes are hidden in the photographs. Since the test image is compared to a database of images that has a face-forwarding picture (two eyes, nose, mouth), there is an erroneous result due to one eye 'missing'. The presence of facial hair on the face also caused inconclusive results because the database image average is an average of 42 face pictures without any facial hair.

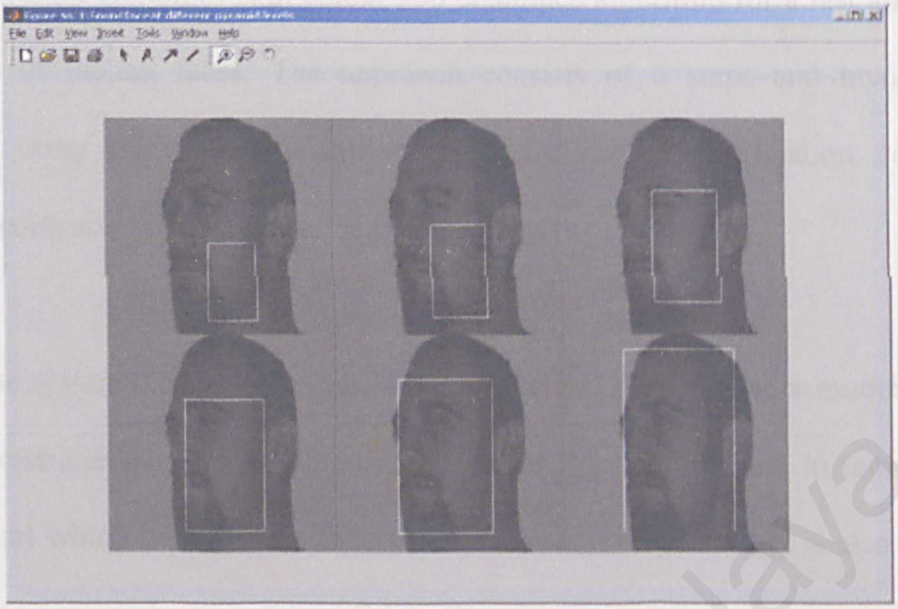


Figure 2.6: Results from Cobb's Face Recognition project

Correct results can be accomplished if the image has the following characteristics:

1) Gray Scale Image

2) Forward Facing Face

3) No obstruction on the face (mask, glasses, facial hair, bandages)

4) No rotation of image

Face Detection System Using Eigenface and Skin Colour [11]

This program combines skin colour and eigenface algorithm for a fast and reliable detection of human faces. The approach consists of 5 steps and involves face detection using skin colour, image segmentation, image classification, edge-based preprocessing and face detection using eigenfaces.

First, the system searches for skin colour region using chrominance model. The hue value is restricted to be less than 0.1 or greater than 0.9. This is to remove some background which has similar colours to human faces. The image is also resize by 1/4 to decrease execution time. This will generate a binary image where white regions represents possible faces. Figure 2.7 shows the face detection result.

Figure 2.7: Skin detection result [11]

The program then looks for region that can be bounded by a rectangle model. Various test are conducted to classify each rectangle region as either face group, non-

face group or face candidate (possible face). Edge-based preprocessing is then conducted on the face candidate. Eyes, nose and mouth produce edges in each face, and this edge mask can be used to remove non-face region. Finally, the eigenface method is used to extract faces from (edge-detected) suspicious rectangles. Figure 2.8 shows the final face detection result.

Figure 2.8: Final face detection result[11]

This program scored around 90% detection rate with 5% false alarm which prove that the combination of eigenface and skin colour is a good method. However, this algorithm relies on face statistics obtained from a small pool of face. photographs. With small number of faces, the diversity, different lightning condition or face rotations cannot be captured.

Face Detection (Using Skin Colour and Template Matching) [13]

This system used a skin colour-based technique to detect frontal human faces in still images. It was built by Henry Chang and Robert Uliyses (Stanford University). The algorithm consist of two image processing step.

The first step is similar to the face detection system discussed in previously (*Face Detection System Using Eigenface and Skin Colour*), whereby skin regions are separated from non-skin regions by using a chroma chart that shows the likelihood of skin colours. The chroma chart generates a gray scale image from the original colour(RGB) image. The gray value at each pixel shows the likelihood of the pixel belonging to the skin. The detected area may not correspond to skin and could be non-face objects that have colour similar to skin colour. Therefore, skin segmentation is done through a thresholding process. The system also takes into account images of different people with different skin colour where an adaptive thresholding process is used to determine the optimal threshold value for skin segmentation.

The second step of the face finder employs facial features to locate face in skin-like segments. The criteria used is the number of holes inside a region where segments with more than one hole is possibly a skin region. Figure 2.9 shows the segmented skin regions using adaptive thresholding as well as a particular skin region selected by the system that corresponds to the input image which is a baby face.

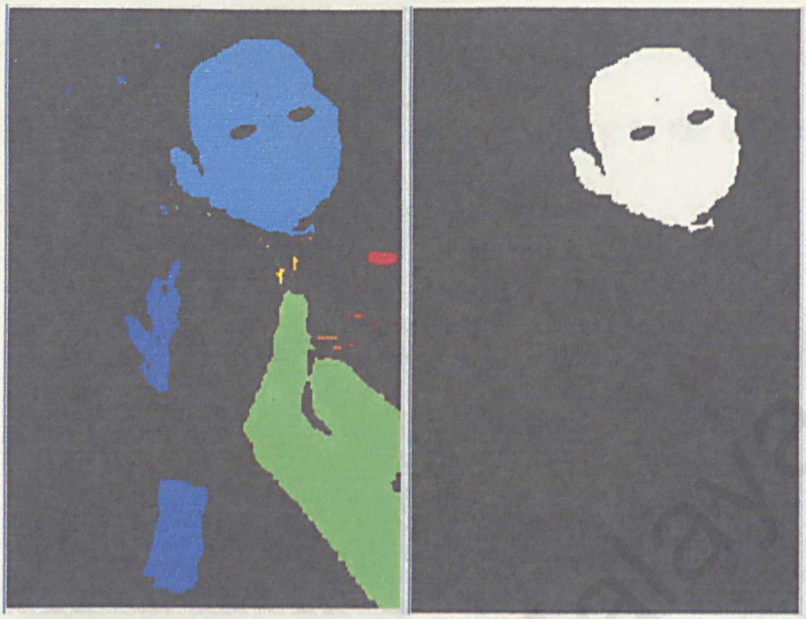


Figure 2.9: (Left) Segmented Skin Regions. (Right) A Skin Region

Then, the characteristics of that particular region is further analyzed. The most important feature of this method is that it uses human face template to make the final decision of determining if a skin region represents a face. Figure 2.10 shows the template used in the system. The template face has to be positioned and rotated in the same coordinate as the skin region image.

Figure 2.10: Template face (model) used to verify the existence of faces in skin regions.

After the system decided that the skin region correspond to a frontal human face, a new image with a hole exactly the size and shape of that of the processed template face is generated. Through computation, the system will yield an image as the original one, but with the template face located in the selected skin region. This is shown in Figure 2.11 and Figure 2.12, in which the face of the baby is replaced by the template face.

Figure 2.11: The face is located.

Figure 2.12: Template face is added.

The coordinates of the part of the image that has the template face is identified. With these coordinates, a rectangle is drawn in the original color image. This is the output of the system which in this case, detected the face of the baby as shown in Figure 2.13.

Figure 2.13: Final result from[13]

2.3 Technology Review

2.3.1 Development Methods

A system development methodology is a well-organized collection of related methods that addresses who does what activities and how, when, why and where these activities should be done to develop a system.

There are many types of development model for software engineering. Most software developers rarely use a single method throughout the system development

process as it may either be insufficient or impractical. The trend is nowadays is to apply a subset of the whole methodology and this subset is called the process framework. However, which model is chosen for development depends on the organization.

Using the Unified Process

Unified Process (UP) is a software engineering process, aimed at guiding software development organizations in their field of work. It uses the object-oriented paradigm. UP is designed and documented using the Unified Modeling Language (UML).

The UP is consists of 4 phases:

1. Inception
2. Elaboration
3. Construction
4. Transition

Within each phase are a number of iterations. Iterations represents a complete development cycle, from requirements capture in analysis to implementation and testing, that results in the release of an executable project

Using Waterfall Model

The waterfall model was the first structured approach to systems development. It gained popularity during the 70s and allowed for a certain amount of order in the

process. The waterfall model is just a time-ordered list of activities to be performed to obtain an IT system.

The activities are listed below:

1. System Conceptualization
2. System Analysis
3. System Design
4. Coding
5. Testing

This model was and remains relatively popular. It strengths are:

- Minimizes planning overhead since it can be done up front.
- Structure minimizes wasted effort, so it works well for technically weak or inexperienced staff.

Unlike prototyping or other methodologies, it does not allow for feedback loops. This means that, for example, once the requirements are written, they should be stable throughout the project. Any reviewing of them is thus an extraneous and unplanned activity.

The waterfall performs well for products with clearly understood requirements or when working with well understood technical tools, architectures and infrastructures. It's weaknesses frequently make it inadvisable when rapid development is needed. In those cases, modified models may be more effective.

Using Spiral Model

The spiral is a risk-reduction oriented model that breaks a software project up into mini-projects, each addressing one or more major risks. After major risks have been addressed, the spiral model terminates as a waterfall model. Spiral iterations involve six steps:

1. Determine objectives, alternatives and constraints.
2. Identify and resolve risks.
3. Evaluate alternatives.
4. Develop the deliverables for that iteration and verify that they are correct.
5. Plan the next iteration.
6. Commit to an approach for the next iteration.

Its strengths include:

- Each iteration of the spiral can be tailored to suit the needs of the project.
- Early iterations of the project are the cheapest, enabling the highest risks to be addressed at the lowest total cost. This ensures that as costs increase, risks decrease.

Its major weakness is that it is complicated and requires attentive and knowledgeable management to apply it. For projects with risky elements, it is beneficial to run a series of risk-reduction iterations which can be followed by a waterfall or other non-risk-based lifecycle.

2.3.2 Operating System

Operating system (OS) is a platform that performs basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

Besides that, the OS makes sure that different programs and users running at the same time do not interfere with each other. For security, OS ensures that unauthorized users do not access the system. OS provides a software platform to allow application programs run on it.

The 2 most commonly-used operating system for PC are Windows operating system and Linux. Below are a brief description of Windows 98 which originates from the Microsoft Windows operating system family, and Linux

Windows 98

Windows 98 is based on the popular Microsoft Windows 95 Operating System, and is designed for the consumer market. Just like Windows 95, Windows 98 was designed for backward compatibility with older DOS and 16 bit programs, as well as providing a platform for the newer (back in 1995) 32 bit programs.

Windows 98 works better by making it simple to access the Internet and by providing better system performance along with easier system diagnostics and

maintenance. With Windows 98, users' system plays better as well with support for the latest graphics, sound, and multimedia technologies, the ability to easily add and remove peripheral devices with support for Universal Serial Bus (USB), and it also enables users to watch TV on PC. Besides that, Windows 98 is compatible with a wide variety of software (including games) and hardware.

Linux

Linux was created by Linus Torvalds at the University of Helsinki in Finland. Unlike Windows 98 and Winodws 2000 Server, Linux can run on a variety of computer architecture. The highlight of Linux is that its source code is freely available to anyone (http://www.linux.org/info/index.html). Most companies will alter the source codes according to their needs.

The key strength of Linux is its security as it found out that the occurrence of virus attack is fewer compared to Windows operating system. Therefore, Linux is often considered as an excellent and low-cost alternative to Microsoft operating system.

However, it is difficult to maintain and administer the Linux operating system. This is one major setback when compared to the easy-to-use Microsoft Windows operating system that has been around for quite some time now. Proper training and knowledgeable experts will be needed in order to manage Linux.

2.3.3 Programming Languages

The question of which programming language to be used to implement the proposed system is very much depend on certain criteria like the type of application that is going to be developed and also the objective of the project. In fact there is an overflowing of programming language in the computing field nowadays and each has its own uniqueness. Thus, careful consideration must be made when choosing the most suitable language. However, many research and experiments has been done on face processing and this has help to identify a few programming languages that has proven to be suitable for implementing a robust system to perform face processing. Two of the most popular language used for implementing face detection are Visual C++ and MATLAB. The suitability of both languages for this project is discussed in the following section.

Visual C++

Visual C++ is part of the Microsoft Visual Studio package. It originates from the C++ language. It is an object-oriented programming language which built programs that consists of classes and functions. Visual C++ supports and provides advanced language extension and the powerful Integrated Developer Environment (IDE) features that enable developers to edit and deploy source codes efficiently. IDE is a completely self-contained environment for creating, compiling, linking and testing Windows programs.

Visual C++ also comes together with the industry standard Active Template Library(ATL) and Microsoft Foundation Classes Library (MFC). The MFC provides

the basis for many of the Windows programs. The MFC is also referred to as an application framework because it provides a set of structured components that provide a ready-made basis for almost any Windows program.

MATLAB

MATLAB is an interactive, matrix-based language for technical and scientific computing, which allows easy implementation of statistical algorithms and numerical simulations. Highlights of MATLAB include the number of toolboxes (collections of programs to address specific sets of problems) available. Hence it has many built-in functions and this enables scientists and engineers to use matrix-based techniques to solve problems without having to write programs in traditional languages such as C or FORTRAN. The MATLAB programming language is exceptionally straightforward to use since almost every data object is assumed to be an array.

The disadvantage of MATLAB is that it tends to use a significant amount of memory and it is relatively slow in executing if-statements, and for- and while-loops that can not be vectorized.

2.4 Chapter Summary

A thorough studies was conducted to seek more information on the few main aspects of Face Detection. Terminologies associated to face detection was explained so that reader will have a better understanding of issues regarding face processing. The most important aspect of this literature was the studies done on the existing methodology/ techniques to detect face. 5 most common method were discussed.

The following section was the technology review which documented the operating system and programming language that can be used to implement the system. This section highlighted the characteristics of the operating systems and languages that were reviewed. It helps in making the decision on which is the most suitable for the Face Detection System.

CHAPTER 3

Methodology

3.1 Software Development Process

A software development process is a method to organize the activities related to creation, delivery, and maintenance of software systems. A Software Development Life Cycle is used to model these processes which consists of a set of necessary activities/phases to develop the software. A method can be classified according to its paradigm. Some of the most common paradigms in software development are structured approach and object-oriented approach. The object-oriented approach is rapidly gaining popularity among software developers. The three most significant software development methods in the object-oriented community are the Booch method by Grady Booch, the Object Modelling Technique (OMT) by Jim Rumbaugh and the Objectory by Ivar Jacobson.

3.2 Methodology Considerations

A methodology is a well-organized collection of related methods that addresses who does what activities and how, when, why and where these activities should be done to develop a system[15]. Due to increased complexity in software nowadays, applying a single method is often insufficient and applying the whole methodology may be impractical. Therefore, software developers will usually opt to apply a subset of the whole methodology and this subset is called the process framework. Also

choosing the right methodology will depend on the system's requirement and its unique operating environment

Although there are many different methodologies, there are fundamental activities which are common to all. They are software specification, software design and implementation, software validation and software evolution.

3.2.1 Benefits of Good Methodology

A good methodology should be adopted for every system development. Below are some of the key benefits of a good methodology.

- The methodology is a proven framework which has uniformity in all aspects of building a software. Hence, user does not need to reinvent the wheel. This framework will serve as a guideline to developers on the required activities and also deliverables at every phase of the development process.

- Defines and focuses roles and responsibilities of different stakeholder. This aids in improving communication among stakeholders as it provides a communication base

- Each method or tool in the methodology results in successful completion of each development task and this is depicted in milestones of the project.

- Problems are addressed early on as reviews procedures are available to identify any errors, inconsistencies and discrepancies during development.

- Improves understanding of user's needs and expectation and encourages the validation of user's needs.

3.2.2 Conclusion on Development Methodology

For the Face Detection System project, the Unified Process (UP) framework which combines the best practices, processes and guidelines of the three object-oriented methods mentioned above will be adopted. It shall be modeled using the Unified Modeling Language (UML) by the Object Management Group (OMG).

3.2.2.1 Overview of the Unified Process (UP)

The Unified Process (UP) is a software life cycle model that uses UML as a modeling tool. UP is regarded as a complete process framework [17] that includes every element in a software development process, including phases, workflows, interactions of workflows, milestone of each phase, activities within each workflow, artifacts or deliverables of each workflow and the visual modeling technique by using UML. Therefore, it is unique when compared to other software development process, namely waterfall model and V-model. These models usually specify only what phases are involved in a software development process and the interactions among these phases.

UP involves phases and workflows. Below are a brief walkthrough of the 4 phases and 5 workflows of UP.

The four phases

The UP consists of the following 4 phases:

- **Inception.** The main goal of inception phase is to establish the business cases for the project. Major activities that are carried out during the inception phase are defining the scope of the system, producing the candidate architecture of the system which is made up of the initial versions of the six primary models of the Unified Process and also estimating the cost, effort and project schedule

- **Elaboration.** The focus of the elaboration phase is to produce the system architecture at the end of it. This system architecture is called the architectural baseline of the system and it contains elements of the expanded versions of the six models initialized during the inception phase.

- **Construction.** The construction phase is concerned with the actual physical development of the system. The system is built iteratively and incrementally during the construction phase. At the end of this phase, a workable beta version of the product should have been built, which cover all the use cases that have been identified during the early stages of development. However the released version is still not final yet and may contains defects.

- **Transition.** This is when the fully functional system is released to its end user. Testing is conducted to trace errors and defects. These errors will be modified and may also include system modification to improve the system.

Within each phase is a number of iterations. Each iteration addresses a set of related use cases or mitigates some of the risks identified at the beginning of the iteration.

The five workflows

The workflows are :

- Requirements
- Analysis
- Design
- Implementation
- Test

The five workflows cut across the four phases and these workflows are iterated within each phase to achieve a certain major milestone of the phase. The completion of each iteration on the other hand marks a minor milestone. This is unlike other traditional development process which models behind a rigid set of phases, namely requirements analysis, system design, implementation and testing.

Architecture of UP

The UP process has two dimensions, as depicted in Figure 3.1. The horizontal dimension represents time and shows the lifecycle aspects of the process as it unfolds. It is express in terms of cycles, phases, iterations and milestones.

The vertical dimension represents core process disciplines (or workflows), which logically group software engineering activities by their nature.

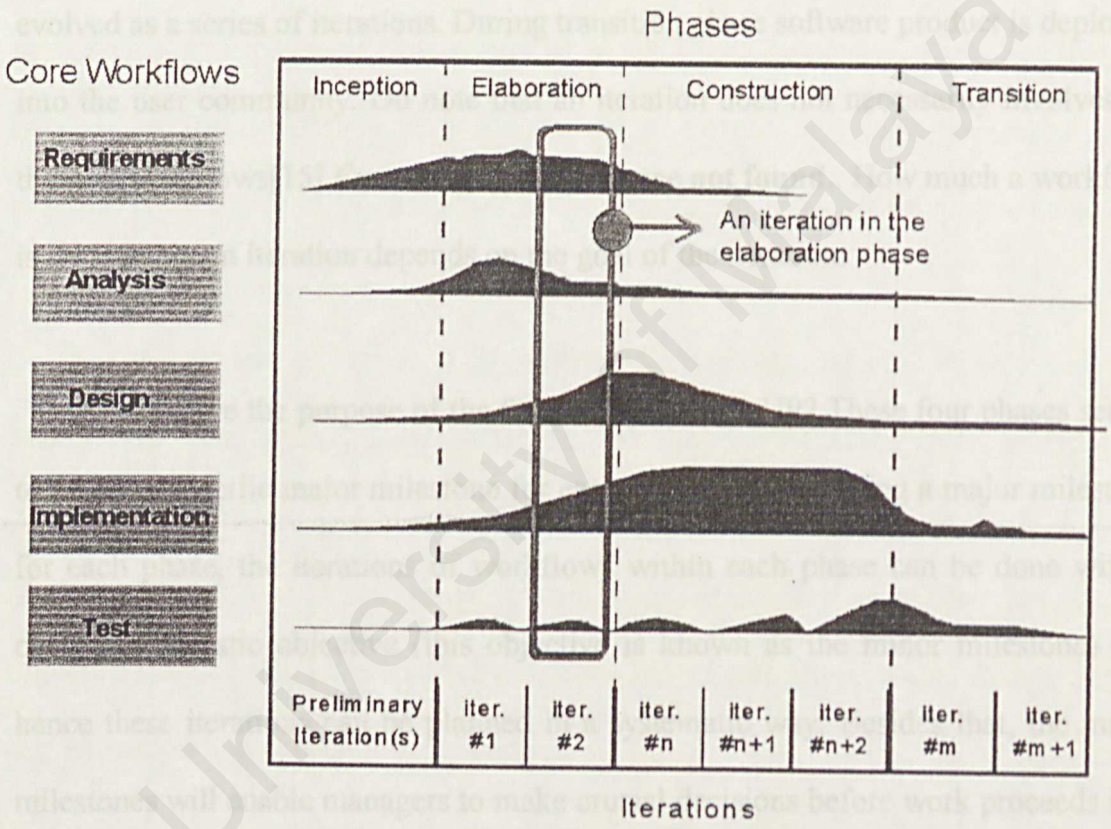


Figure 3.1: The architecture of Unified Process [16]

As shown in the Figure 3.1, during elaboration phase, the requirements and analysis activities are allocated most of the resources and also during the construction phase, the resource requirements for requirements and analysis activities

diminish, but the design and implementation activities are allocated more resources. A typical iteration goes through all the five workflows as shown also in Figure 3.1.

During inception phase, the problem is defined and actors and use-cases are identified. In the elaboration phase, most of the product's use-cases are specified in detail and the system architecture is designed. In addition, project plan is developed during this phase. The purpose of the construction phase is to develop a software product which is ready to be introduced into the user community. The product is evolved as a series of iterations. During transition phase software product is deployed into the user community. Do note that an iteration does not necessarily involves all the five workflows[15] **Error! Reference source not found.**. How much a workflow is involved in an iteration depends on the goal of the iteration.

What are the purpose of the four new phases in UP? These four phases serves to define a specific major milestone for each phase. By specifying a major milestone for each phase, the iterations of workflows within each phase can be done with a clear and realistic objective (this objective is known as the minor milestone) and hence these iterations can be planned in a systematic way. Besides that, the major milestones will enable managers to make crucial decisions before work proceeds into the next phase and also to monitor the progress of work in each of the four phases.

3.2.2.2 The Three Unique Characteristics of the Unified Process

There are three important characteristics of the Unified Process - use case driven, architecture-centric, and iterative and incremental. These characteristics outlines the uniqueness of UP compared to other software development processes.

The Unified Process Is Use Case Driven

The Unified Process is a use case driven process due to the fact that the system analysis, design, implementation and test process or workflow are based on use cases. Before discussing further on use cases, some associated terminologies should be defined first: actor and use case. An actor is the user or an external system that interacts with the current system in order to achieve a particular result from the system. A use case on the other hand is a sequence of actions that the system performs in order to yield that particular result to the actor. Use cases are used to capture all the functional requirements of a system and they are represented collectively in the use case model. Workflows are also derived from the use cases.

The use case model serves as a guideline for software developers to create the analysis and the design model. After all, the function of the analysis and design model is to realize and specify the use cases. The use case model is then implemented based the implementation model. Finally, test is carried out on the implementation to validate that all use cases have been implemented and to verify that all use cases are implemented correctly.

The Unified Process Is Architecture-Centric

A software architecture embodies the most important static and dynamic aspects of the system. It gives developers a view of the whole design with the most important characteristics more visible by leaving the details aside. In this sense, the Unified Process is an architecture-centric software development process. In the Unified

Process, the functional capabilities of a software is driven by use cases and the architecture of the software must allow enough room for the realization of these use cases, for now and in the future. The architecture of the system in the Unified Process is often finalized at the end of the elaboration phase. It can only be identified by understanding the key or crucial use cases of the use case model. These use cases constitute the core system functions. Therefore there is often an interplay between use cases and the architecture. Both must evolve simultaneously.

The Unified Process Is Iterative and Incremental

The Unified Process supports the iterative and incremental nature of software development. The Unified Process is made up of five workflows which iterates continuously until the final product is released to the customer. In each iteration, a minor milestone is defined. If the minor milestone is achieved, then the software development process will continue with the next iteration. If it is not, the developers will have to revisit their previous decisions and try a new approach. In this way, if anything goes wrong, the organization only suffers losses from the misdirected effort of that particular iteration or mini project, and not the value of the entire project. Besides that, by splitting a project into a few mini projects or iterations, more realistic short term goals can be set to each iteration. Developers will hence have more confidence in achieving these clear and short term goals, rather than a long and unrealistic goal in other non-iterative software development processes.

3.2.2.3 The Primary Models of the Unified Process [16]

There are six primary models in Unified Process. These models are constructed to visualize and specify the system at different viewpoints to different group of people. Each model in the Unified Process is composed of a set of diagrams that are documented by using the Unified Modeling Language (UML). The UML diagrams shall be explained in Section 3.4.1. The following text provides a brief description of the six primary models in the Unified Process.

Use Case Model

A use case model describes what the system has to do for its users. It is made up of one or more UML use case diagrams. The use case model is built during the requirements workflow. It is also extremely important in organizing and modeling the behaviors of a system.

Analysis Model

The analysis model is constructed to describe the use cases precisely and also to structure them in a way that they can be understood. An analysis model is usually composed of the UML class diagrams and interaction diagrams. It can also have activity diagrams and state chart diagrams if necessary. The analysis model is constructed during the analysis workflow.

Design Model

The design model describes the physical realization of use cases and focuses on how functional and nonfunctional requirements are shaped up together with other constraints that are related to the implementation environment. The design model is very important as it is an abstraction of the implementation of the system, where a well-defined design model can be simply mapped into implementation. A design model usually consists of the UML class diagrams, interaction diagrams and activity diagrams. However the class diagrams and interaction diagrams in the design model are explained in greater detail and they involve more objects, compared to their counterparts in the analysis model. The design model is constructed during the design workflow.

Deployment Model

The deployment model describes the physical distribution of the system in terms of computational nodes. It specifies where a component of the system should be located on the network. It also depicts the mapping between the software architecture and the hardware architecture (the combination of both architecture is known as the system architecture). A deployment model is made up of one or more UML deployment diagrams and they are constructed during the design workflow.

Implementation Model

The implementation model maps the elements in the design model, such as design classes and packages, which are implemented in terms of software components, such

as source codes files and executable files. For example, a class can be implemented in Microsoft Visual Basic as a physical file with the extension .cls. An implementation model consists of one or more UML component diagrams and they are established during the implementation workflow.

Test Model

The Unified Process' test model presented in textual form, and not graphical form. A test model includes test cases, test procedures and test components. They are all used to how unit, integration and system testing are carried out. A test case is a document that specifies what to test with which input or result and under what condition to test. A test procedure on the other hand specifies how to perform one or more test cases. Finally a test component automates one or more test procedures. It can be developed using scripting or programming language.

3.2.2.4 Justification of Methodology

The following are the reasons why the Unified Process is chosen as the software development process framework for Face Detection System:

- Unified Process (UP) supports the complete software development life cycle, from analysis phase to design and right up to the programming phase where coding are done. In contrast, the procedural paradigm only supports the design and programming steps since a potential solution must be specified before task decomposition can begin.

- As UP designs are inherently more maintainable. This advantage is due to the fact that UP as an object-oriented paradigm forces designers to create encapsulated objects. Encapsulation means that the interface of the object is separated from the implementation of the object. Such separation is highly desirable because many parts of the program may depend upon a certain interface. If changes must be made to the interface, then widespread changes will also have to be made to the program. A good object oriented design will minimize the interface changes that must be made. Hence, maintenance is localized to the implementation of a certain object.

- The Unified Process comes along with a highly rated Unified Modeling Language (UML). The UML offers highly-descriptive notations that are easy to understand and can model all system requirements and specifications. Besides providing the standard notations, UML allows software developer to extend the language by introducing stereotypes, constraints and tagged values. Before the introduction of UML, different methods and modeling technique are used for different models. For example, the Booch method only covers the analysis and design phase of software development. In order to construct implementation and testing models, some other methods will have to be used. In order to tie these 'different' models together, a complex technique of transition and mapping will be required. This will not only slow down the software development process but also incur errors introduced in moving from one modeling technique to another.

- UP supports the reuse of software more than structured analysis, through inheritance. Inheritance itself *is* reuse, the defining of new classes as extensions

of previously defined classes. This is applicable to the Face Detection System as the Microsoft Foundation Classes(MFC) will be used as the development tool and numerous pre-defined MFC classes will be used.

- UP is iterative and incremental in nature and thus it is very suitable for developing software. Modern software development is iterative in nature because of uncertainty in many areas during the initial stage, namely user requirements. So naturally it is subjected to changes in later stages. Changes to user requirements or mistakes in the development process simply mean that the software development process will have to be iterated.

3.3 Unified Modeling Language

The Unified Modeling Language (UML) is "a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems"[18]. It is founded on the work of Ivar Jacobson (OOSE – Object Oriented Software Engineering), Grady Booch (The Booch Method) and James Rumbaugh (OMT – Object Modeling Technique) [5]. UML has a very rich collection of notations, which can be used to model any static and dynamic aspect of the system. In case that the standard UML notations fail to address some of the aspects of the system, UML allows users to extend the language itself by using the three constructs of stereotypes, tagged values and constraints. The primary UML artifacts are shown in Figure 3.2. It illustrates the diagrams needed when the system is large and complex.

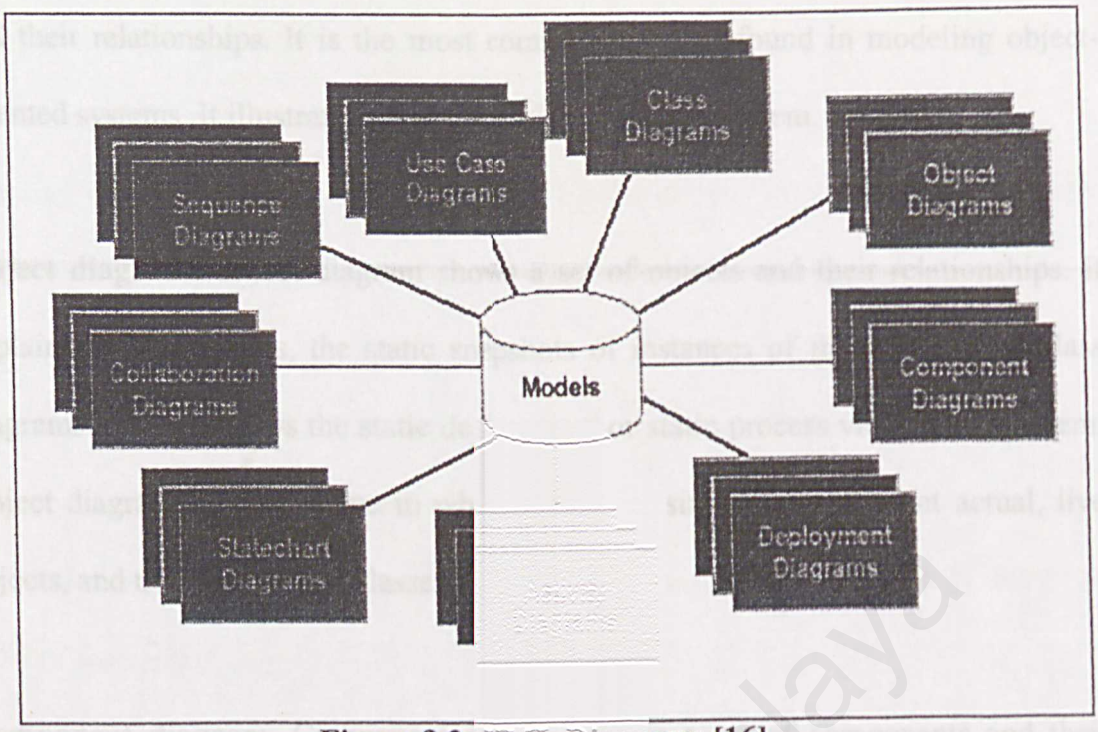


Figure 3.2: UML Diagrams [16]

3.3.1 The Unified Modeling Language (UML) Diagrams

The following text gives a brief and general description of all the UML diagrams. It is beyond the scope of this text to explain every feature of the UML as it will take up hundreds pages of documentation. However, further information on UML can be found in the book entitled *The Unified Modeling Language Reference Manual* by James Rumbaugh, Ivar Jacobson and Grady Booch (Addison-Wesley, 1999).

Use case diagram. Use case diagram shows a visualization of the relationship between actors and use cases. It illustrates the static use-case view of the system. Also, it is important in organizing and modeling the behaviors of the system

Class diagram. Class diagram shows a set of classes, interfaces, and collaborations and their relationships. It is the most common diagram found in modeling object-oriented systems. It illustrates the static design view of a system.

Object diagram. Object diagram shows a set of objects and their relationships. It explains data structures, the static snapshots of instances of things found in class diagrams. Also, it shows the static design view or static process view of the system. Object diagrams have a scope in which they live, since they represent actual, live objects, and not just abstract classes

Component diagram. Component diagram shows a set of components and their relationships. It illustrates the static implementation view of the system. Component diagrams are related to class diagrams in that a component is typically maps to one or more classes, interfaces, or collaborations.

Deployment diagram. Deployment diagram is a set of nodes and their relationships. It explains the static deployment view of architecture. These diagrams are related to component in that a node is typically encloses one or more components.

Activity diagram. Activity diagram displays the flow from activity to activity within a system. An activity shows a set of activities, the sequential or branching flow from activity to activity, and objects that act and are acted upon. It illustrates the dynamic view of a system. Activity diagrams are important in modeling the function of a system. Moreover it emphasizes the flow of control among objects.

Statechart diagram. Statechart diagram shows a state machine, consisting of states, transitions and events, and activities. It illustrates the dynamic view of a system. It is important while modeling the behavior of an interface, class, or collaboration. In addition, it emphasizes the event-ordered behavior of an object, which is especially useful in modeling reactive systems.

Collaboration diagram. Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects connected by communication links. It illustrates the dynamic view of a system.

Sequence diagram. The UML sequence diagram is an interaction diagram that focuses on the time ordering of messages that go back and forth between objects, in order to realize a use case. It shows exactly the same information as the collaboration diagram. However the emphasis of both diagrams is different as the collaboration diagrams emphasizes on the structural organization of objects whereas the sequence diagram focuses on timing of messages. Between the two, sequence diagram is more commonly used. This is because the primary concern in system analysis is on finding requirements and responsibilities of objects and not finding detailed and chronological sequences of interactions.

3.4 Information Gathering Methods

A proper and effective method for gathering information is vital to ensure a sound understanding of the system in all aspects. These acquired information will

eventually serve as a groundwork for determining the requirements of the system and also for designing the system. As every system is unique in its nature, there is rarely a fixed set of rules on methods to gather information. In fact there are general methods, for example reviewing hard data like written documents or reports, interviewing, observation and sampling, which cut across system type and are modified so as to be more applicable to the proposed system. For the proposed Face Detection System, the method used for gathering information are describe below:

Internet research

The internet is a vast resource for information in basically every field. As this project is related to computer vision and image processing, research on the internet involves researching on the various technique for face detection and its corresponding algorithm, which are published as journals and papers. Moreover, University of Malaya is a paying subscriber to the IEEE database, so students can use the resources for free.

Books and references

Referring to books and printed references is also another way of gathering information. The books are obtained from University of Malaya's main library. Our faculty, FSKTM, also has a document room where seniors' thesis reports are stored. In fact thesis done former undergraduates and masters students proves to be very good source of information.

Informal Interview and Questionnaire

Interviews and questionnaire are direct methods to gather information. A total of 40 people were interviewed and at the end of the interview session, they were asked to fill up a questionnaire (see Appendix A). The interviews were carried out informally as the main objective is to find out about the user's perception of a face detection system. The questionnaire actually complements the interview whereby the objective is to find out about user's preference in a face detection system. It is one way to elicit user requirement for the Face Detection System. This shall be further discussed in Section 4.1: Techniques for Requirements Elicitation.

Discussion with supervisor

Consultation session with supervisor was held from time to time to discuss issues about the proposed system. As Face Detection System was taken up for educational purposes, there is no problem statement. In the absence of a problem statement, the supervisor needs to be consulted to determine the direction of the project. Moreover, the discussion also serves as informal interview as the supervisor for this project, Puan Azwina Md. Yusof, has had experience in developing a face recognition system.

3.5 Conclusion On Tools and Technology

Having reviewed and analyzed the available technologies and tools, the most suitable and appropriate tools for developing the system are identified and selected. The selected tools include the programming language and the operating system.

Selected programming language

Visual C++ 6.0 wil be used to implement the Face Detection Sysetm. It will handle the user interface design and also the functionalities of the system. Below are the reasons why Visual C++ was chosen:

- Visual C++ includes the comprehensive Microsoft Foundation Classes(MFC). MFC has pre-defined classes for building Windows applications, so it accelerates as well as simplifies the development process. Moreover, it is able to support the mathematical manipulations required in implementing the proposed face detection algorithm.

- Visual C++ was chosen over MATLAB even though MATLAB provides a vast library of pre-defined image processing functions. In fact that is *the* reason for not selecting MATLAB. When building a system which relies heavily on pre-defined functions will limit the programmer's control of the overall system's functionalities. Hence, using Visual C++ which requires most of the image processing functions to be coded from scratch, is the better choice.

- Visual C++ supports object-oriented technology whereby different classes could be developed to handle different aspects of the face detection algorithm. This will provide much help in the design and development of the system in order to solve the face detection problem.

- Visual C++ provides the facilities that aid in the design of user interface as well as the Face Detection system as a whole. This is possible because Visual C++ includes sophisticated resource editors that support the design of complex textboxes, menus, toolbars, images and many more.

- Visual C++ is able to present the graphical views of the applications structures as it is being developed. This feature is supported by Visual C++'s excellent integrated development environment called the Develop Studios.

- Visual C++ supports an integrated debugging tool. It allows every aspect of the system that is being developed to be examined in minute detail as it is run. This will provide much needed aid when debugging programs in order to examine any errors found.

Selected operating system

The Face Detection System will be implemented under the Windows 98 operating system. Deciding on which platform / operating system as the system platform is straightforward. This is due to the fact that Windows 98 has proven to be able to perform in a stable condition. Besides, it provides a user-friendly way to interact with the system through icons, compare to Linux.

3.6 Chapter Summary

By choosing and adhering to a good software development process, this project aims to produce a system of commendable quality. Unified process will serve as a guideline as to what activities should be done at what stage. This will ensure a systematic way of developing the proposed system.

Information gathering also plays an important role to this project. The information gathered are not only about the technologies and techniques. It also includes gathering information on the requirements.

At the last section, a conclusion was made as to what programming language and operating system will be most suitable for the proposed system.

CHAPTER 4

System Analysis

4.1 Techniques For Requirements Elicitation

Capturing requirements of a system, or in a more formal note eliciting requirements, seems to be a very straightforward process. However, it does face several barriers which complicates the whole process. The problems are many and one of it could be that the user do not know what they want, or that the analyst think they understand user problems better than users do. Nevertheless, there are techniques that have proved effective in addressing these problems.

For this project, some information gathering methods mentioned in the previous chapter were in fact requirements elicitation techniques used to capture requirements for the proposed Face Detection System.

Informal interview and questionnaires

Interviews were conducted in an informal way which means there is no fixed set of questions. Basically, the objective of the interviews was to learn about the public's perception of a face detection system. At the end of the interview session, user were asked to fill up a questionnaire. The purpose of the questionnaire is to learn about user's preference and expectation regarding the systems' features. By establishing a feature set, more specific requirements can be defined [19]

The users' response during the interviews as well as from the questionnaires were analyzed. The analysis yield a list of preferred and expected features of the system. From this set of features a list of requirements was specified and documented (please refer Section 4.2 for details). The next step was to get supervisor/lecturer's agreement of the features and validation of the requirements. It is found out that majority find the concept of providing 'human-like' vision to computers very interesting. Majority also tend to be confused about the unique characteristics of a face detection system and other face processing system like face recognition. After analyzing the response to the question asked in the questionnaire, This users' misconception was addressed promptly in order to prevent capturing the wrong requirements.

Use Cases

Another elicitation technique used to capture the requirements of the Face Detection System is applying use cases. Use cases provide a simple, structured format to define the behavior of the proposed system[19]. In fact, they served to counter check stakeholder's acceptance of the possible external design and to elicit fine detail of functionality.

4.2 System Requirement Analysis

A system requirement is a description of the needs and desires for an information system or software. Conformance or lack of conformance to a set of requirements often determines the success or failure of a project. Therefore, it is important to ensure an in-depth understanding of the requirements of a system before design takes

place. System requirements can be further classified into functional requirements and non-functional requirements.

4.2.1 Functional Requirements

Functional requirements are the statements of services the system should provide, how the system should react to a particular input, and how the system should behave in a particular situation. The functional requirements of the Face Detection System are as below. They are also modeled using use cases which is shown in Figure 4.1.

- **Load image as input**

  The system shall receive as input an image or picture file from the user. User shall be able to load the image from a folder or file. This image shall be loaded into the system and used as a source for the subsequent face detection task.

- **Detect human face(s) in input image**

  The system shall be able to detect the presence of human face(s) contained in the input image. The detected human face(s) will be automatically marked with rectangle on the original image. The system will redisplay/reproduced this image as output.

- **Perform image processing**

  The system shall preprocess the input image before the execution of the face detection algorithm. The image processing shall include converting the image into a pre-defined format and also into the required image representation (gray-

scale image, binary image, etc.). For example, converting a 24-bit per pixel image into an 8 bit-per pixel image.

- **Display images generated at different stages**

  The system shall be able to display the different types of image generated at different stages of the face detection process. The system shall provide the means for user to select which image to be viewed

- **Perform mathematical computations**

  The system must be able to perform all the mathematical calculations required by the face detection algorithm. The system must be able to generate accurate results for every calculation.

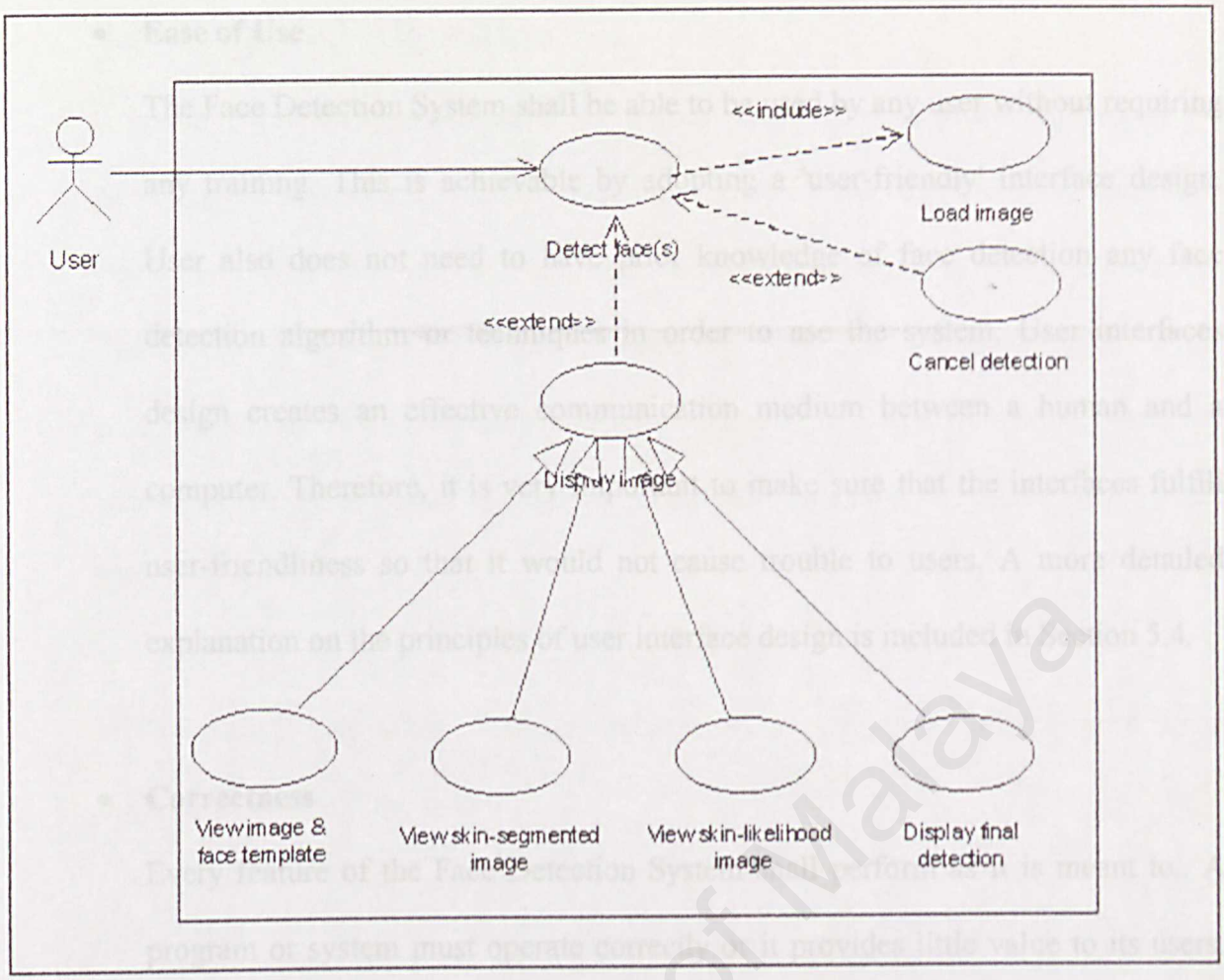


Figure 4.1: Use case diagram for Face Detection System

4.2.2 Non-Functional Requirements

Non-functional requirements refers to the constraints on the services or functions offered by the system.

- **Reliability**

  Reliability is the extent to which a program can be expected to perform its intended function with precision. The Face Detection System shall be able to yield good face detection accuracy with minimal false detection.

- **Ease of Use**

  The Face Detection System shall be able to be used by any user without requiring any training. This is achievable by adopting a 'user-friendly' interface design. User also does not need to have prior knowledge of face detection any face detection algorithm or techniques in order to use the system. User interfaces design creates an effective communication medium between a human and a computer. Therefore, it is very important to make sure that the interfaces fulfill user-friendliness so that it would not cause trouble to users. A more detailed explanation on the principles of user interface design is included in Section 5.4.

- **Correctness**

  Every feature of the Face Detection System shall perform as it is meant to.. A program or system must operate correctly or it provides little value to its users. Correctness is the degree to which the software performs its required function. To ensure the quality of the Face Detection System, extensive testing and trial-and-errors will be carried out before the system is rolled out..

- **Maintainability**

  Maintainability is the ease with which a program can be corrected if an error is encountered, adapted if its environment changes, or enhanced if the customer desires a change in requirements. As the Face Detection System will be built using Visual C++ which enhances the object-oriented concept, therefore, the system shall be easily maintained.

4.3 Proposed Algorithm

Having done an extensive study on the existing technique and methodology for detecting faces in an image, a designated algorithm for the Face Detection System is finally proposed. The proposed algorithm is a combination of skin colour approach and template matching. The algorithm starts with segmenting the skin regions in the input image and then performing analysis on these regions to search for skin regions that constitute to a frontal human face. Template matching is applied to the latter stage.

This project also takes other researches work as guideline and reference when building developing the system.

4.3.2 Justification of chosen technique

The decision to adopt the skin colour approach stems from a number of simple but powerful characteristics of skin colour. Firstly, processing skin colour is simpler than processing any other feature. Secondly, under certain lighting conditions, colour is orientation invariant. To reduce false detection, the template matching technique is incorporated to the algorithm. The result is a simple yet robust algorithm to detect face.

4.3.3 Overview of the proposed algorithm

The steps involved in the proposed algorithm are depicted in Figure 4.2. Every step is further explained in the following text.

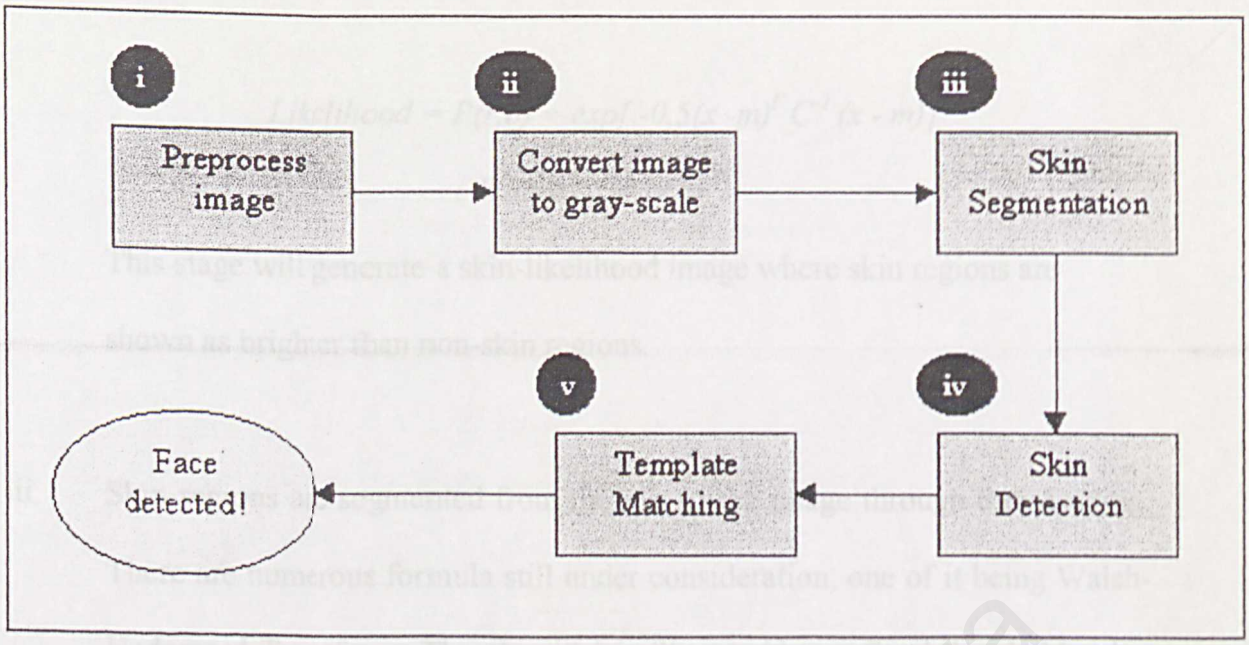


Figure 4.2: Proposed algorithm for face detection

i. The first step is to preprocess the input image. When the image is loaded, the system will convert the 24 bit per pixel image into an 8 bit per pixel image and store this new image as a vector of n x m length, where n and m are the dimensions of the image. Then this vector will be converted into a matrix of n x m dimension.

ii. The image will undergo a normalization process to transform it into gray-scale representation, or better known as monochrome image. The formula used is [20]:

$$r = R/(R+G+B)$$

$$b = B/(R+G+B)$$

note: (R,G,B) is the colour pixel vector for red, green and blue

The gray value of every pixel will be used to calculate its likelihood to be a skin pixel. The formula is as follow [20]:

$$Likelihood = P(r,b) = exp[\ -0.5(x - m)^T C^{-1} (x - m)]$$

$$Where : x = (r,b)^T.$$

This stage will generate a skin-likelihood image where skin regions are shown as brighter than non-skin regions.

iii. Skin regions are segmented from the rest of the image through thresholding. There are numerous formula still under consideration, one of it being Walsh-Hadamard Transform. The algorithm will use either a fixed threshold value or adopt the adaptive thresholding approach where the threshold value will be adjusted and the corresponding segmented region size is observed. The threshold value that results in a minimum increase in skin region will be taken as the optimal threshold value. This step will yield a binary image which present the skin areas as white and others as black.

iv. Skin region will be studied to determine if that region constitute to a face. The skin region will be studied in terms of number of holes, center of mass, orientation, width and height of the region and the region ratio. These are actually the required parameters to do template matching in subsequent step. Below are the mathematical algorithm to compute some of the parameters:

- The skin region can be short-listed by counting number of holes in the region (the holes could be eyes or mouth). The number of holes can be computed using the Euler [22] number, defined as follow:

$$E = C - H$$

Where E is Euler number,
C: the number of connected components
H: the number of holes in a region

- To determine the center of the region /mass[22]

$$\bar{x} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} jB[i,j]$$

$$\bar{y} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} iB[i,j]$$

Where: B is the matrix of size[n x m] representation of the region
A is the area in pixels of the region

- To compute the axis for rotated /tilted face [22]

$$\textit{Angle of inclination, } \theta = 1/2 \; atan \frac{b}{a - c}$$

where:

$$a = \sum_{i=1}^{n}\sum_{j=1}^{m} (x'ij)^2 B[i,j]$$

$$b = 2\sum_{i=1}^{n}\sum_{j=1}^{m} x'ijx'ijB[i,j]$$

$$c = \sum_{i=1}^{n}\sum_{j=1}^{m} (y'ij)^2 B[i,j]$$

and:

$$x' = x - \bar{x}$$
$$y' = y - \bar{y}$$

The other 2 parameters- width and height of region, and region ratio are pre-defined values.

v. The system will search for regions whose shape matches with the pre-defined template (an ellipse). The matching is done by computing the cross-relational value between the part of the image corresponding to the skin region and the template. The coordinates of the region that matches the template will be obtain. Using this coordinate, a rectangle will be drawn on the original image to indicate detected face.

4.4 Hardware requirements

Table 4.1 shows the minimum hardware requirements for the Face Detection System.

Table 4.1 : Hardware requirements

	Face Detection System
Processor	Intel Pentium or compatible 166Mhz.
Memory	64MB
Disk Space	Not known yet
CD-ROM Drive	Needed for installation
Keyboard & Mouse	Yes

4.5 Software Requirements

There are 2 types, one is the software(s) for developing the overall system, the other one is those needed to in order to run final system. For this purpose, the required software will be MS Visual C++, Rational Rose 2000 Enterprise Edition and Microsoft words for documentation. The software required to run the Face Detection system is just one which is the Windows 98 operating system

4.6 Chapter Summary

This chapter mainly deals with the requirements of the system. At the beginning it was explained what technique were used to elicit the system requirements. After that a list of functional and non-functional requirements were specified. Besides the minimum hardware and software requirements for the system was also mentioned.

The algorithm proposed to detect face was also explained so as to have a clear picture about what steps were involved in detecting faces. The algorithm was chosen based on existing techniques.

CHAPTER 5

System Design

In the previous chapter of System Analysis, we identified the requirements of the system, in other words, WHAT needs to be in the proposed system. In System Design, it will be determine HOW it will be done. Therefore the objective of system design is to detail the requirements that have been identified during system analysis, ad using these requirements as a basis to plan the structure of the system which will be implemented. This section shall concerned with the overall architecture of the system and the setting of standards in standards in terms of user interface.

5.1 Design of the System Functions

All the functionalities of the Face Detection System are encompassed within one module. This module deals with all aspects of the system, which are the face detection and preprocessing of image (sub-function). It uses the skin colour and template matching approach to solve the face detection problem.

It takes as input an image loaded by user. This is the image upon which the system needs to detect the presence of human faces. Once the image has been acquired, the process of detecting face can begin. However, before starting the detection process, the input image will be preprocessed whereby it will be converted into the required format in order to perform the detection process. This shall be further explained in the next section. This detection process is done by executing a face detection

algorithm. Every detected face will be marked with a rectangle on the original image. This will be displayed as the output. If there is no detected face, the output will be just the original image without any markings.

This module also handles the option of viewing the different images generated at different stages of the face detection process. These images are actually the original input image in different image representation, for example binary and gray-scale.

Both functions of detecting face and viewing optional image has been identified as functional requirements of the Face Detection System during requirements analysis and modeled as use cases (please refer Figure 4.1). Figure 5.1 shows the sequence diagram for *detect face use case*. Figure 5.2 shows the sequence diagram for the *display image use case* whereby the user select to view the skin-segmented image.

An activity diagram in Figure 5.3 is used to model the flow of processes for the whole of face detection system.

Knowing the process involved in this module besides having a clear picture of the functions of this module, will facilitate the design of other aspect of the system and the effective implementation of the system.

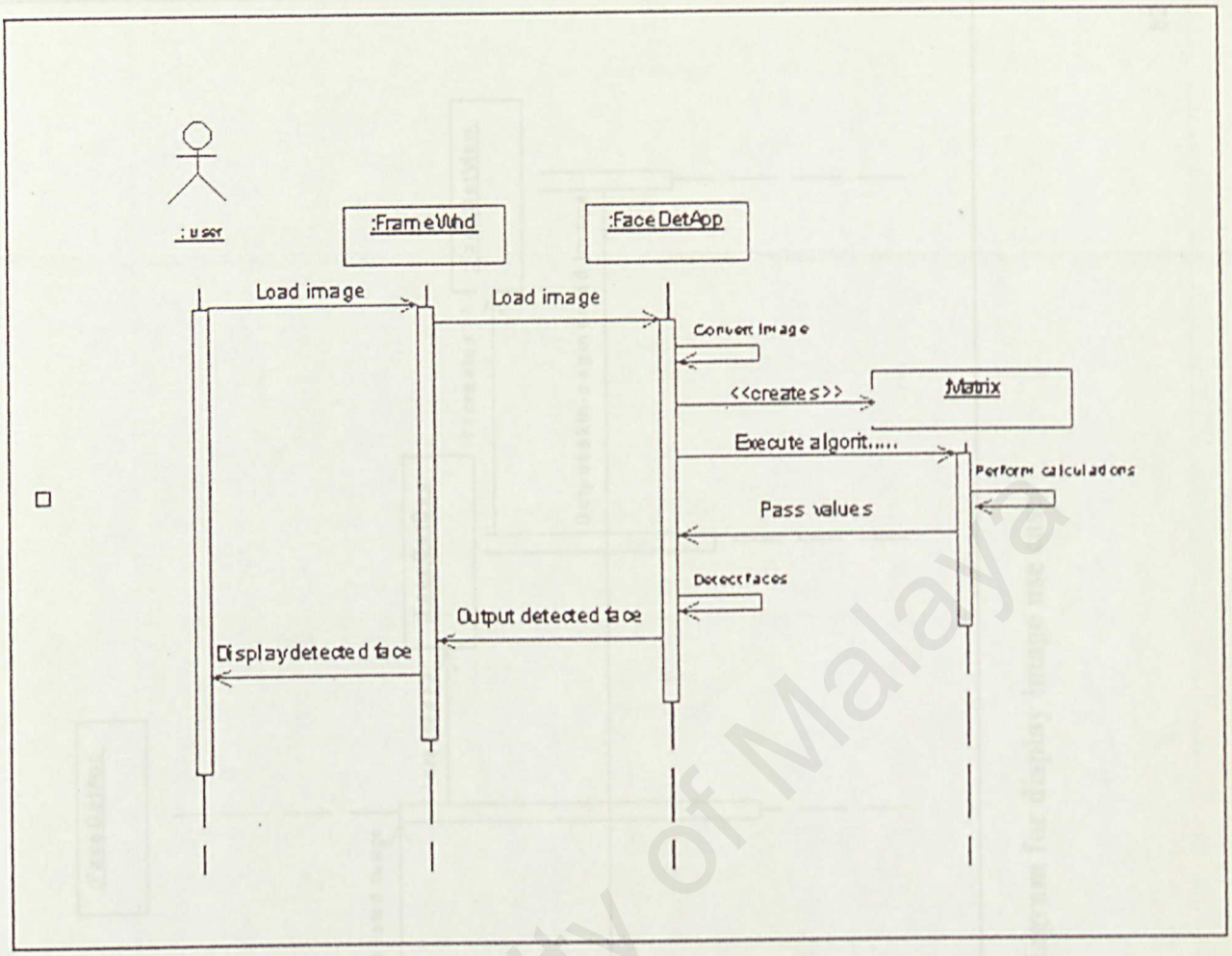


Figure 5.1: Sequence diagram for detect face use case

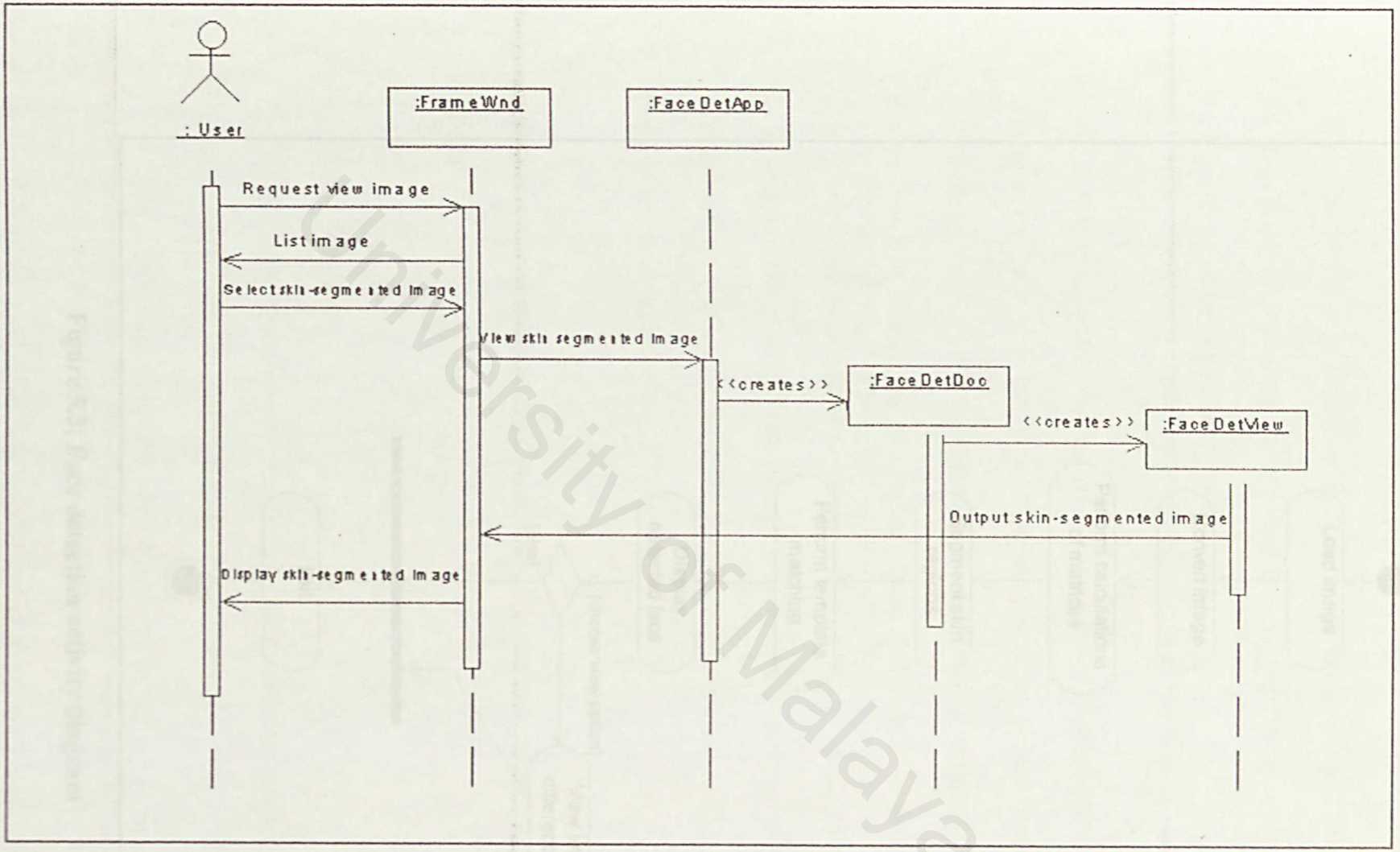


Figure 5.2 : Sequence diagram for display image use case

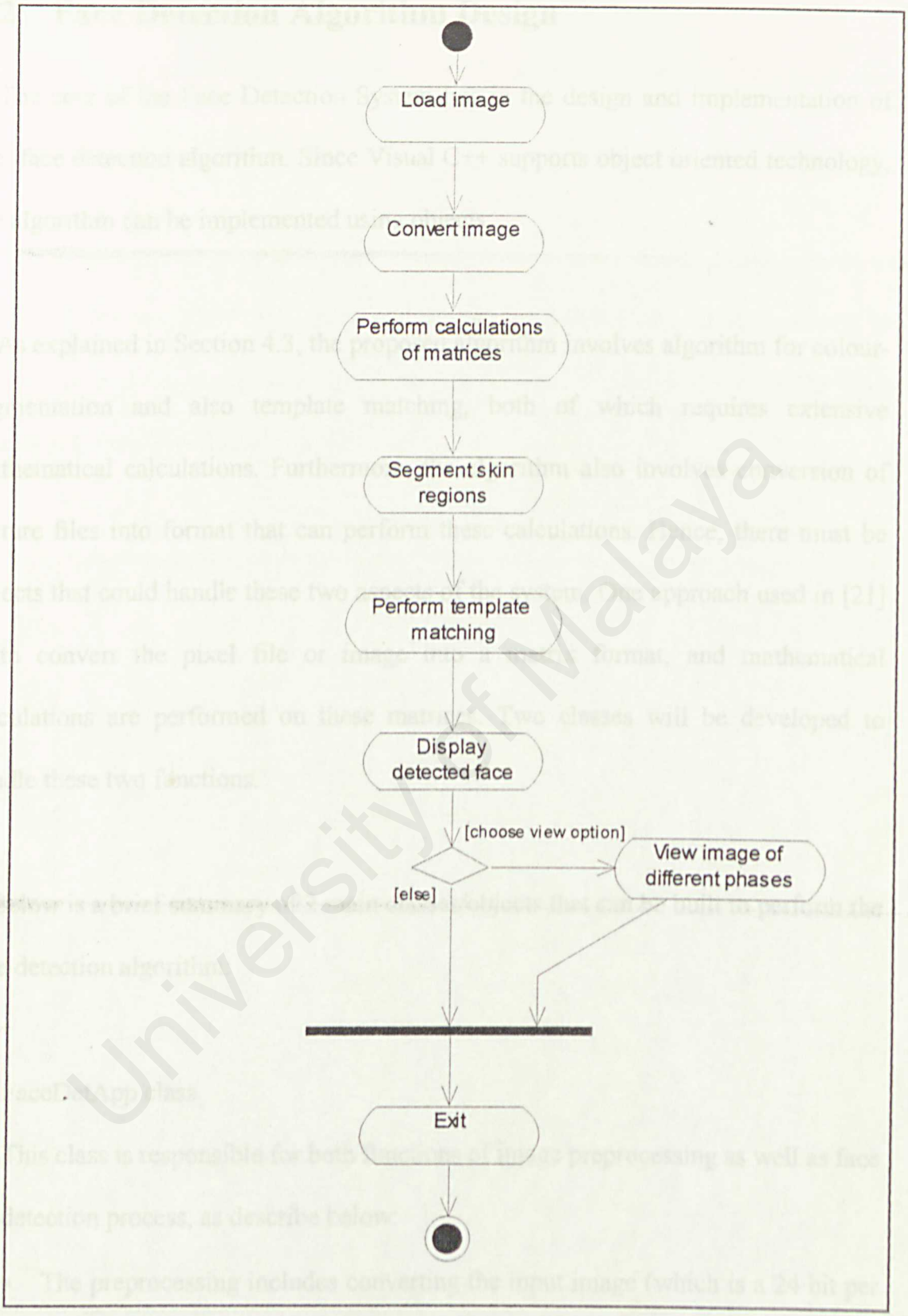


Figure 5.3: Face detection activity diagram

5.2 Face Detection Algorithm Design

The core of the Face Detection System lies in the design and implementation of the face detection algorithm. Since Visual C++ supports object oriented technology, the algorithm can be implemented using objects.

As explained in Section 4.3, the proposed algorithm involves algorithm for colour-segmentation and also template matching, both of which requires extensive mathematical calculations. Furthermore, the algorithm also involves conversion of picture files into format that can perform these calculations. Hence, there must be objects that could handle these two aspects of the system. One approach used in [21] is to convert the pixel file or image into a matrix format, and mathematical calculations are performed on these matrices. Two classes will be developed to handle these two functions.

Below is a brief summary of 2 main classes/objects that can be built to perform the face detection algorithm:

a) FaceDetApp class

This class is responsible for both functions of image preprocessing as well as face detection process, as describe below:

- The preprocessing includes converting the input image (which is a 24 bit per pixel image) into an 8 bit per pixel image and storing this new image as a vector of n x m length, and then representing the pixels matrices.
- This class is also responsible for all other functions of face detection, namely segmenting skin region, thresholding , template matching, etc.

b) Matrix class

This class is responsible for handling mathematical calculations of matrices such which includes adding, subtracting, and multiplying.

In addition to the 2 proposed main classes, several pre-defined classes which are provided by the MFC will be used. These pre-defined classes are mainly for designing and managing the user interface design. Figure 5.4 shows the class diagram for the Face Detection System.

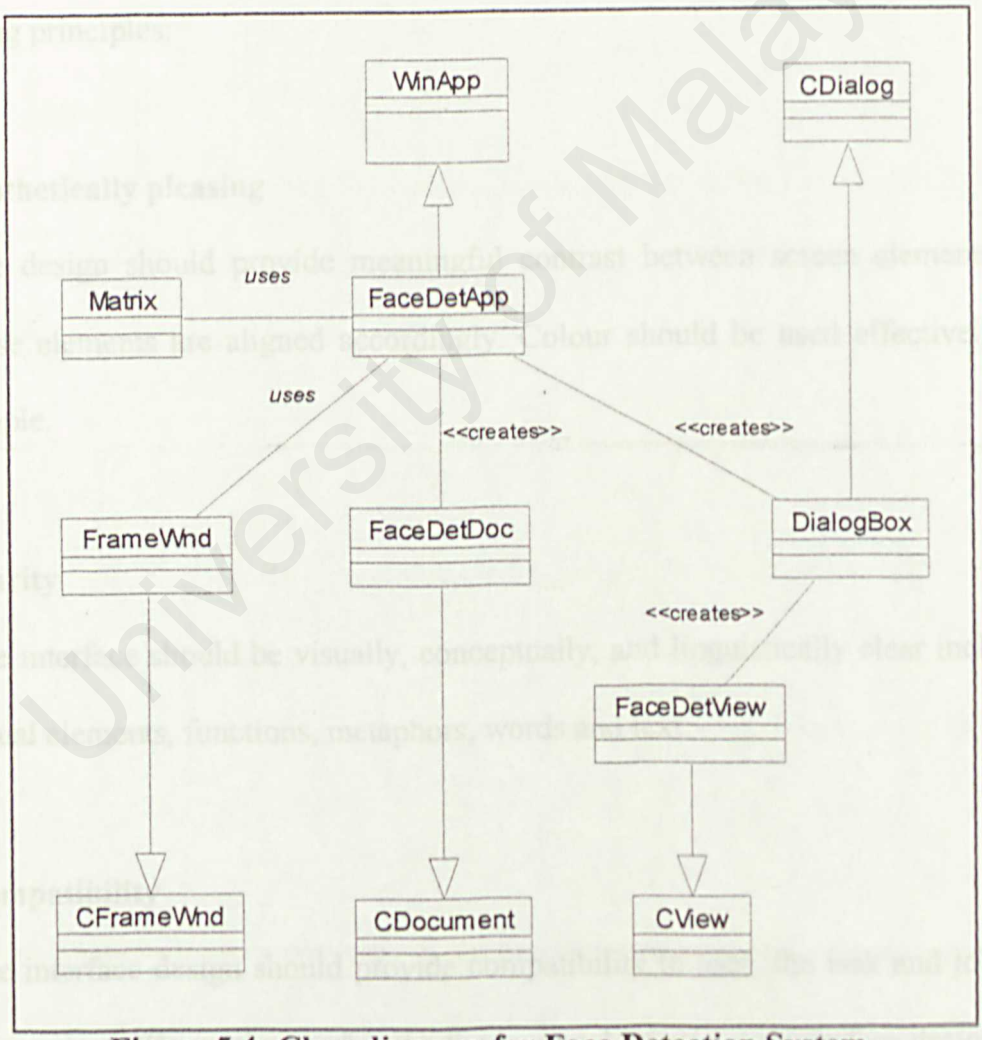


Figure 5.4: Class diagram for Face Detection System

5.3 User Interface Design

A user interface design is a collection of techniques and mechanisms to interact with something. The usability of the Face Detection System depends heavily on the user interface design. A good user interface design will reduce the time to learn to understand and use the system. It will also increase the productivity of users as tasks can be performed faster with less error.

5.3.1 Adopted principles

The user interface for the Face Detection System is designed based on adhere to the folowing principles:

- **Aesthetically pleasing**

  The design should provide meaningful contrast between screen elements and these elements are aligned accordingly. Colour should be used effectively and simple.

- **Clarity**

  The interface should be visually, conceptually, and linguistically clear including visual elements, functions, metaphors, words and text.

- **Compatibility**

  The interface design should provide compatibility to user, the task and job, and the product. "Know the user" is the fundamental principle in interface design.

- **Comprehensibility**

  A system should be easily learned and understood. *A user should know what to look at, what to do, when to do, why to do and how to do.* The flow of actions, responses, visual presentations and information should be in a sensible order that is easy to recollect and place in context.

- **Consistency**

  A system should look, act, and operate the same throughout. For example, the same action should always yield the same results.

- **Transparency**

  Permit the user to focus on the task or job, without concern for the mechanics of the interface. Workings and reminders of workings inside the computer should be invisible to the user.

- **Flexibility**

  Flexibility is the system's capability to respond to the individual differences in people. People should be able to interact with a system in terms of their own particular needs, including knowledge , experience, and personal preference.

- **Familarity**

  Concerns employing familiar concepts and use a language that is familiar to the user. The interface shall be kept natural, mimicking the user's behaviour patterns.

5.3.2 User Interface for Face Detection System

Figure 5.4 is the layout for the user interface design of the system. However the proposed interface will be subjected to changes if necessary, during the implementation phase.

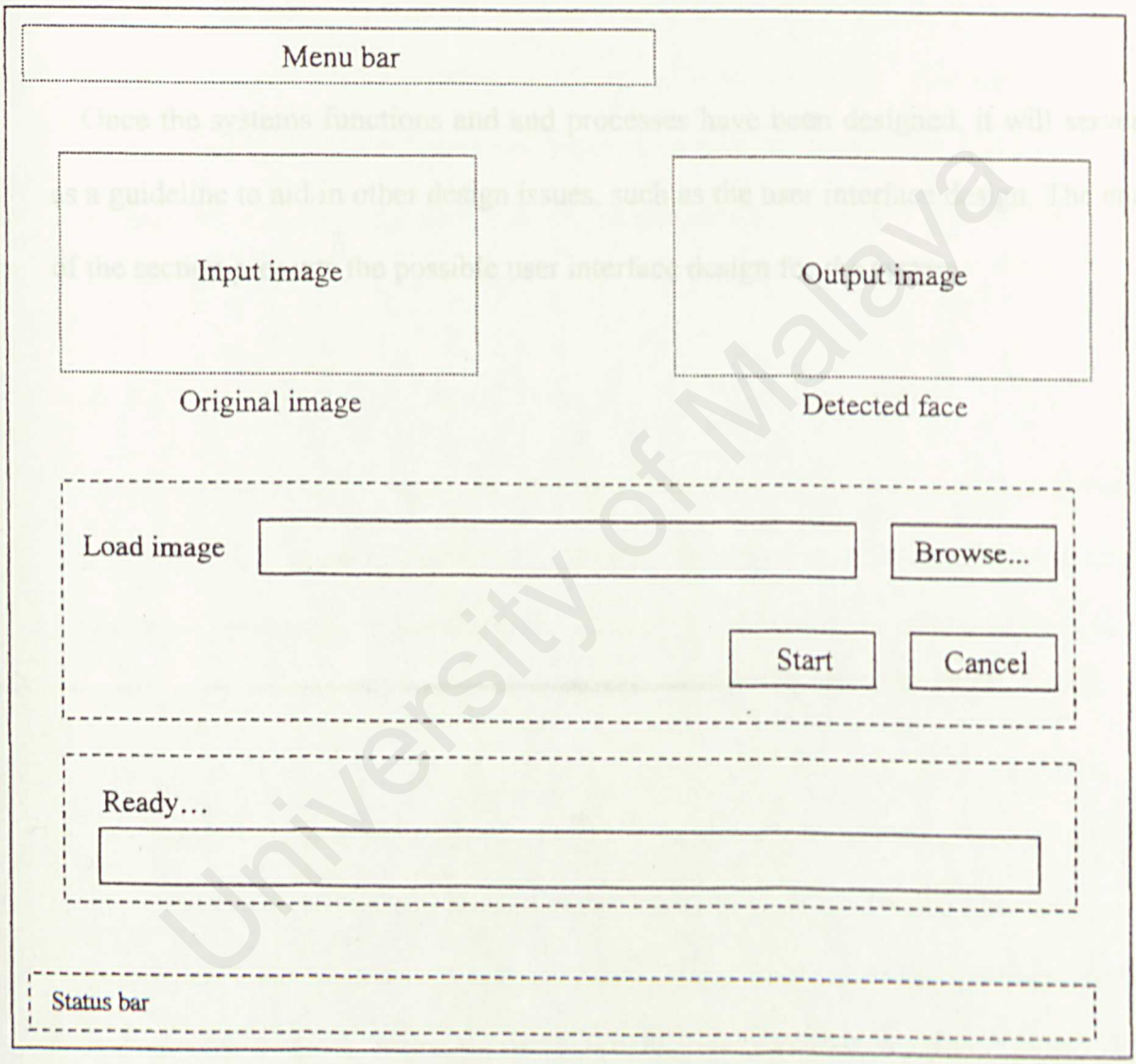


Figure 5.5: Layout of system's user interface design

5.4 Chapter Summary

System design provides a bridge between system analysis and system implementation. In this chapter, the functionalities of the system were outlined and ways of designing the system in order to be able to perform those functionalities were discussed.

Once the systems functions and and processes have been designed, it will served as a guideline to aid in other design issues, such as the user interface design. The end of the section presents the possible user interface design for the system.

CHAPTER 6

System Implementation

The Face Detection System is implemented in Visual Basic 6.0. This is different from what was initially proposed in Chapter 3 whereby it was proposed that the system shall be developed using MS Visual C++ 6.0. The reasons for the change in the programming language used shall be discussed later at the end of this chapter.

6.1 Processing the Input Image

An image is actually made up of pixels Therefore any image processing task (including skin segmentation or generally face detection) will require the system to be able to extract the properties/values of every pixel in order to manipulate/process each pixel.

In chapter 4, it was proposed that the input image will be pre-processed whereby once loaded the system will convert the 24 bit per pixel image into an 8 bit per pixel image and store this new image as a vector of n x m length, and then this vector will be converted into a matrix of n x m dimension. However, in using VB6.0 to implement the Face Detection System, the image need not undergo such preprocessing. Instead, every pixel is stored in the form of a dynamic array where the value for each pixel is stored according to its (x,y)coordinate.

2 functions adapted form the Windows library **gdi32** was used in the face detection task. They are:

i) **GetPixel (hdc** *hdc*, **int** *X* , **int** *Y*) :

The function retrieves the red, green, blue (RGB) color value of the pixel at the specified coordinates(x,y). The return value is the RGB value of the pixel.

ii) **SetPixel (hdc** *hdc*, **int** *X* , **int** *Y*, **colorref** *crcolour*)

The function sets the pixel at the specified coordinates to the specified colour. *X* and *Y* specifies the x-coordinate and y-coordinate respectively of the point to be set while crcolour specifies the colour to paint the point.

Both functions are widely used in the coding especially in converting input images to different form of images (gray scale, binary, chromatic space) and also to retrieve the value of every pixel to do comparison with a face template as part of the template matching process.

The following coding depicts the system as it sets the value of every pixel of the input image to the calculated skin probability value:

*SetPixel PictureA(3).hdc, x, y, RGB(Int(Probability * 255), Int(Probability * 255), Int(Probability * 255))*

The following coding is to retrieve every pixel value of the template face (*FaceTemplate75.hdc*) and store them in an array (*FaceTemplate75Array(xBase, yBase)*)

```
For xBase = 0 To 74
   For yBase = 0 To 99
      FaceTemplate75Array(xBase, yBase) = GetPixel(FaceTemplate75.hdc, xBase,
yBase) And 255
   Next yBase
Next xBase
```

6.2 Implementation of Algorithm

As proposed in Chapter 4, the algorithm used for detecting human faces in the Face Detection System consists of 2 main parts based on the techniques of skin segmentation and template matching.

The algorithm starts with segmenting the skin regions in the input image and then performing analysis on these regions to search for skin regions. Template matching is applied to the latter stage where every skin region is searched in order to find regions which are similar to a pre-defined face template. The algorithm and its corresponding source codes is detailed below.

6.2.1 Skin segmentation

The first step of the program is to take the original color image and convert the colors into chromatic color ("pure color") space in order to eliminnate lighting effects (luminance component). The format of the RGB space (original image) includes luminance, which makes it difficult to characterize skin colors because lighting effects can change the appearance of the skin. There are numerous ways to convert an image from RGB to chromatic colours. The method used in this system is :

If $(RedLevel_{x,y}+BlueLevel_{x,y}+GreenLevel_{x,y}) > 0$:

- $CRedLevel_{x,y}=(RedLevel_{x,y}/(RedLevel_{x,y}+BlueLevel_{x,y}+GreenLevel_{x,y})) \times 255$
- $CBlueLevel_{x,y}=(BlueLevel_{x,y}/(RedLevel_{x,y}+BlueLevel_{x,y}+GreenLevel_{x,y})) \times 255$
- $CGreenLevel_{x,y}=(GreenLevel_{x,y}/(RedLevel_{x,y}+BlueLevel_{x,y}+GreenLevel_{x,y}))\times 255$

Otherwise, if $(RedLevel_{x,y}+BlueLevel_{x,y}+GreenLevel_{x,y}) = 0$:

- CRedLevel = 0
- CBlueLevel = 0
- CGreenLevel = 0

After the chromatic conversion, we then use the following algorithm to determine the probable skin region.

$$\textbf{Likelihood} = P(r,b) = \exp[-0.5(x-m)^T C^{-1} (x-m)]$$

Input pixel: $x = \begin{pmatrix} r \\ b \end{pmatrix}$

Mean: $m = \frac{1}{n}\left(\sum \begin{pmatrix} r \\ b \end{pmatrix}\right)$

Covariance: $C = \frac{1}{n}\left(\sum (x-m)(x-m)^T\right)$

As the scope of this system / project does not encompasses the building of its own skin model, the values used for the calculation of skin likelihood is based on the statistical analysis done by [24] to generate the mean and covariances of the chromatic red and chromatic blue components of skin coloured pixels. The following values are the results of the statistical analysis, and the final P(r,b) equation gives the final probability of a pixel representing skin.

Both the chromatic conversion and calculation of skin probability are depicted in the following source code:

```
For y = 0 To 239
    For x = 0 To 319
        PColor = GetPixel(PictureInput.hdc, x, y)
        RedLevel = PColor And 255
        GreenLevel = (PColor And 65280) / 256
        BlueLevel = (PColor And 16711680) / 65536
        CheckLevel = (RedLevel + GreenLevel + BlueLevel)
        If CheckLevel = 0 Then CheckLevel = 1
        CRedLevel = (RedLevel / CheckLevel) * 255
        CGreenLevel = (GreenLevel / CheckLevel) * 255
        CBlueLevel = (BlueLevel / CheckLevel) * 255
        Probability = Exp((-0.017395259992469 * CRedLevel * CRedLevel) _
                + (5.93467379379089 * CRedLevel) _
                - (3.12761747829197E-02 * CRedLevel * CBlueLevel) _
                + (5.56756031786492 * CBlueLevel) _
                - (0.015980450780927 * CBlueLevel * CBlueLevel) _
                - (513.200963758186))
```

After determining the probability of every pixel of the image, a skin likelihood image (also known as grayscale image) giving the probability based on the gray level, as depicted below:

```
SetPixel PictureA(3).hdc, x, y, RGB(Int(Probability * 255), Int(Probability * 255),
Int(Probability* 255))
```

After getting a grayscale image of skin likelihood, it is then necessary to threshold the image into a binary image. The purpose is to segment the probable skin regions (shown as white region in the binary image) from the rest of the image/background (the black region). It is based on the formula [26]:

If A[x,y] ≥ 0 **A[x,y] = object =1**
Else **A[x,y] = background = 0**

To date there is no universal procedure for threshold selection that is guaranteed to work on all images as different images require different threshold. Hence it needs to be chosen by hand. In this Face Detection System, the threshold can be adjusted by

user during run time through a slider control. The threshold process is as depicted below:

```
Threshold = SldThreshold.Value
If Probability >= Threshold Then
        SetPixel PictureA(2).hdc, x, y, RGB(255, 255, 255)
Else
        SetPixel PictureA(2).hdc, x, y, 0
End If
```

6.2.2 Template matching

It is now necessary to evaluate each segmented skin region to determine if it is a face. The program uses a modified "window-box" search method, and then compares the candidate faces to a template face.

First, the system will search for segmented skin regions which are shaped like an oval shape. As illustrated in Figure 6.1 below, at least 4 of the white pixels and all four of the black pixels must be found in a certain search box before it will be considered a candidate face. This pattern eliminates all of the skin regions which are not shaped like an oval.

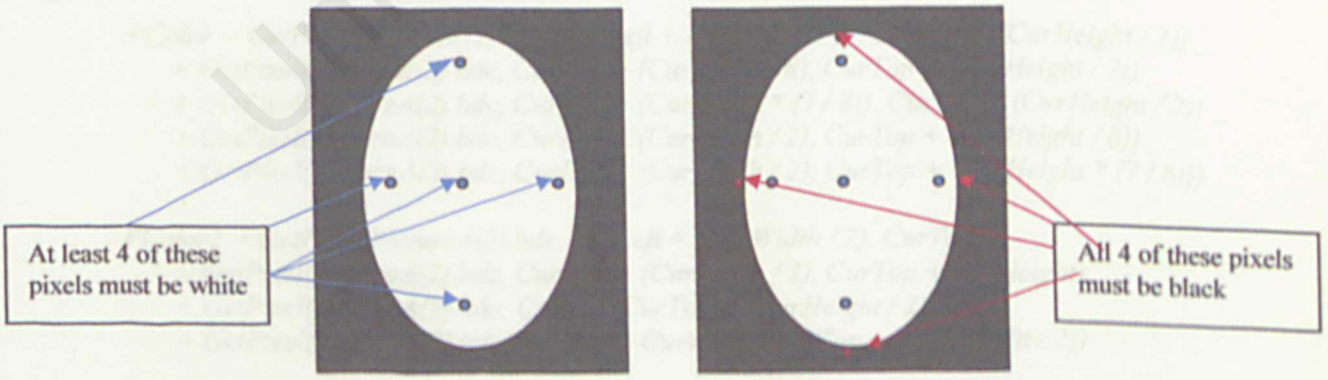


Figure 6.1: The required pixels and their positions to pass off as oval shape.

This search stage begins with the largest "window" possible; that is, an image height search box with the aspect ratio of 3:4 (height to width). It moves the search window left to right (and top to bottom, if possible) until no more windows are left to check at that size. Then, the program shrinks the search window and repeats the left-to-right, top-to-bottom search pattern. Also, a colored box is drawn around each candidate face in the binary skin map. The colour of the box represents the degree of probability of the "boxed region" to be a face. Green box represents the highest probability, blue represents medium probability and light blue represents lowest probability. The candidate face positions are then stored in an array.

The following source code depicts the algorithm for searching oval-shaped skin region. First, the pixel value of the pixels at the 4 black pixel position and 5 white pixel position are retrieved from the input image and stored. As the input images are not processed beforehand, it is difficult to have face regions which fulfils exactly the requirement of 4 black pixels and 5 white pixels as mentioned above.

```
Do While ((CurTop + CurHeight) <= PictureA(1).ScaleHeight)
  Do While ((CurLeft + CurWidth) <= PictureA(1).ScaleWidth)

    PColor = GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 2), CurTop + (CurHeight / 2)) _
      + GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 8), CurTop + (CurHeight / 2)) _
      + GetPixel(PictureA(2).hdc, CurLeft + (CurWidth * (7 / 8)), CurTop + (CurHeight / 2)) _
      + GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 2), CurTop + (CurHeight / 8)) _
      + GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 2), CurTop + (CurHeight * (7 / 8)))

    PColor2 = GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 2), CurTop) _
      + GetPixel(PictureA(2).hdc, CurLeft + (CurWidth / 2), CurTop + CurHeight) _
      + GetPixel(PictureA(2).hdc, CurLeft, CurTop + (CurHeight / 2)) _
      + GetPixel(PictureA(2).hdc, CurLeft + CurWidth, CurTop + (CurHeight / 2))

    If PColor >= (RGB(255, 255, 255) * 5) And PColor2 = (RGB(0, 0, 0) * 4) Then
      Debug.Print PColor & " and " & PColor2 _
        & " at (" & CurLeft & "," & CurTop & ")-(" _
        & (CurLeft + CurWidth) & "," & (CurTop + CurHeight) & ")"
      SetPixel PictureA(1).hdc, CurLeft + (CurWidth / 2), CurTop + (CurHeight / 2),
RGB(255, 0, 0)
```

As mentioned above the following source code will then store the face candidate positions as array:

```
DoEvents

PFXPosHi(UBound(PFXPosHi)) = CurLeft
PFYPosHi(UBound(PFYPosHi)) = CurTop
PFHeightHi(UBound(PFHeightHi)) = CurHeight
PFWidthHi(UBound(PFWidthHi)) = CurWidth
PFFaceHi(UBound(PFFaceHi)) = CurFace
ReDim Preserve PFXPosHi(UBound(PFXPosHi) + 1)
ReDim Preserve PFYPosHi(UBound(PFYPosHi) + 1)
ReDim Preserve PFHeightHi(UBound(PFHeightHi) + 1)
ReDim Preserve PFWidthHi(UBound(PFWidthHi) + 1)
ReDim Preserve PFFaceHi(UBound(PFFaceHi) + 1)

PFXPosAll(UBound(PFXPosAll)) = CurLeft
PFYPosAll(UBound(PFYPosAll)) = CurTop
PFHeightAll(UBound(PFHeightAll)) = CurHeight
PFWidthAll(UBound(PFWidthAll)) = CurWidth
PFFaceAll(UBound(PFFaceAll)) = CurFace
ReDim Preserve PFXPosAll(UBound(PFXPosAll) + 1)
ReDim Preserve PFYPosAll(UBound(PFYPosAll) + 1)
ReDim Preserve PFHeightAll(UBound(PFHeightAll) + 1)
ReDim Preserve PFWidthAll(UBound(PFWidthAll) + 1)
ReDim Preserve PFFaceAll(UBound(PFFaceAll) + 1)
```

Through testing we have found that the three best combinations of black pixels and white pixels to correspond to a oval shaped will be

i) 5 or more white pixels and 4 black pixels at the respective positions mentioned above (highest possibility of face candidate)

ii) 4 or more white pixels and 4 black pixels at the respective positions mentioned above (moderate possibility of face candidate)

iii) 5 or more white or more white pixels and at least 3 black pixels at the respective positions mentioned above (low possibility of face candidate)

After all of the search windows have been checked and all candidate face positions have been stored, the program compares each candidate face with the template face. The template face used is from an eigenface face recognition system. The reason for another template face matching process after the oval-shape matching is because it is not accurate enough to just pass every oval-shaped skin region as human face. Therefore the purpose of this final matching stage is to analyse every oval-shape skin region to determine if it has black holes at positions that corresponds to that of facial features like eyes, nostrils and mouth. Finally, a green box is drawn around every detected face.

Figure 6.2 shows the template face used in the final stage of template face matching.

Figure 6.2: Template face

The final stage of matching the oval-shaped face candidates with a template face is depicted in the following source code:

```
For ySub = CurTop To (CurTop + CurHeight) Step (CurHeight / 100)
    x = 0
    For xSub = CurLeft To (CurLeft + CurWidth) Step (CurWidth / 75)
      SetPixel TempOutPic.hdc, x, y, GetPixel(PictureA(0).hdc, xSub, ySub)
      x = x + 1
    Next xSub
    y = y + 1
  Next ySub
  CurFace = 0
  For y = 0 To 99
    For x = 0 To 74
      GrayLevel = GetPixel(TempOutPic.hdc, x, y) And 255
      CurFace = CurFace + (GrayLevel - FaceTemplate75Array(x, y))
    Next x
  Next y
  CurFace = Abs(CurFace)
```

```
.
.
.
For i = 0 To 9
    MinFaceDiff = 99999999999#
    MinFaceIndex = -1
    For j = 0 To UBound(PFXPosAll) - 1
      If MinFaceDiff > PFFaceAll(j) Then
        MinFaceIndex = j
        MinFaceDiff = PFFaceAll(j)
      End If
    Next j

    y = 0
    If MinFaceDiff <> 9999999 Then

      For ySub = PFYPosAll(MinFaceIndex) To (PFYPosAll(MinFaceIndex) +
PFHeightAll(MinFaceIndex)) Step (PFHeightAll(MinFaceIndex) / 100)
        x = 0
        For xSub = PFXPosAll(MinFaceIndex) To (PFXPosAll(MinFaceIndex) +
PFWidthAll(MinFaceIndex)) Step (PFWidthAll(MinFaceIndex) / 75)
          x = x + 1
        Next xSub
        y = y + 1
      Next ySub
      PictureA(0).Line (PFXPosAll(MinFaceIndex), PFYPosAll(MinFaceIndex))-
(PFXPosAll(MinFaceIndex) + PFWidthAll(MinFaceIndex), PFYPosAll(MinFaceIndex)),
RGB(0, 255, 0)

      PictureA(0).Line (PFXPosAll(MinFaceIndex), PFYPosAll(MinFaceIndex))-
(PFXPosAll(MinFaceIndex), PFYPosAll(MinFaceIndex) + PFHeightAll(MinFaceIndex)),
RGB(0, 255, 0)

      PictureA(0).Line (PFXPosAll(MinFaceIndex), PFYPosAll(MinFaceIndex) +
PFHeightAll(MinFaceIndex))-(PFXPosAll(MinFaceIndex) + PFWidthAll(MinFaceIndex),
PFYPosAll(MinFaceIndex) + PFHeightAll(MinFaceIndex)), RGB(0, 255, 0)

      PictureA(0).Line (PFXPosAll(MinFaceIndex) + PFWidthAll(MinFaceIndex),
PFYPosAll(MinFaceIndex))-(PFXPosAll(MinFaceIndex) + PFWidthAll(MinFaceIndex),
PFYPosAll(MinFaceIndex) + PFHeightAll(MinFaceIndex)), RGB(0, 255, 0)

    End If
      DoEvents
    PFFaceAll(MinFaceIndex) = 9999999
  Next i
```

6.3 Implementation of the System Design and Functionalities

The functions in the Face Detection System are programmed according to the type of control that calls or triggers the function. In other words, the codings for different functions are 'classified' under its corresponding control objects (command button, slider, etc.). For example, the codings to perform skin segmentation are written under the object CmdSegment which is a command button object. The procedure will be executed when user click on the CmdSegment command button, hence the procedure which triggers the skin segmentation is the "click-event" on the object. This is as depicted in the source code below:

```
Private Sub CmdSegment_Click()
   Dim x As Integer, y As Integer
   Dim PColor As Long
   Dim RedLevel As Integer, GreenLevel As Integer, BlueLevel As Integer, GrayLevel As
Integer
   Dim CRedLevel As Single, CGreenLevel As Single, CBlueLevel As Single, CGrayLevel As
Single
   Dim CheckLevel As Integer
   Dim PrevGrayLevel As Integer, MaxDiffLevel As Integer, Threshold As Single
   Dim Probability As Double
   PrevGrayLevel = 128
   MaxDiffLevel = 0
   Threshold = SldThreshold.Value / SldThreshold.Max
   .
   .
   .
End Sub
```

All global variables of the program are declared separately under a separate module file. The declaration of the GetPixel function and SetPixel function are also declared in this module file. This is to make it more convenient to call these functions and variables as they are used extensively throughout the program.

6.4 Discussion

As opposed to what was initially planned at the earlier phase of this project, the Face Detection System was implemented in using Visual Basic 6.0 instead of Visual C++ 6.0. The main reason for the change is because it is relatively more convenient to do complex image manipulation with the use of a picture box object. Moreover, there are also some inevitable changes in the implementation of the system design from what was proposed due to not implementing the system in Visual C++. However, the concept of developing the system using the object-oriented approach has not been compromised. As stated in Section 6.3, every function of the system is programmed according to the control object, and this objects are members of a particular class For example the command button class defines the properties, events and methods that all member of the command button class support.

There has also been some improvision on the face detection algorithm in the template matching part. Initially it was proposed that the template matching would would match face template based on the number of holes in the segmented face candidate as well as the centre of mass. It was proposed then that the holes will be computed using Euler number and the centre of mass of every skin region is computed so that the template face can be resized and rotated accordingly in order to see if it fits nicely on the region. However, when implementing the system, it was found that this method of template matching is very difficult and tedious to be implemented without having a library of certain pre-defined functions like in Matlab. Therefore, instead of searching for holes and centre of mass, the implemented template matching searches for oval-shaped skin region. This serves as an early elimination of non-face candidate before the system compares the face candidate to a template face.

6.5 Chapter Summary

This chapter explains in detail how the algorithm for skin segmentation and template matching, which are methods used in the Face Detection System to detect face, is implemented and coded in Visual Basic 6.0. There have been some changes in the system implementation compared to the initial project proposal. One being the programming language used to implement the system. The reason for implementing the system in Visual Basic 6.0 instead of Visual C++ is explained in section 6.4 Discussion. The proposed algorithm for face detection was also improvised. Despite these differences the system was implemented successfully.

CHAPTER 7

Testing

The Face Detection System was put through different phases of testing in order to ensure that it fulfills the functional and nonfunctional requirements as well as it is bugs free. The testing phase of the Face Detection system is made up of three levels – unit, integration and performance testing. Unit and integration testing are white box testing techniques and they are normally carried out concurrently with the implementation of the system. When any error or bugs were detected during the unit and integration testing, an inspection and debugging of the code had to be done. Performance testing on the other hand is a black box testing technique, where the testing is only concerned with whether the application has fulfilled the functional and nonfunctional requirements as planned, and well as the accuracy of the system in detecting faces.

7.1 Unit Testing

Unit testing is conducted on the Face Detection System to point out 2 general errors - algorithmic and computational. Algorithmic and computational errors are errors that are caused by wrong processing steps in a function or subroutine, hence producing the wrong output to a particular input. The unit testing phase has helped to discover and solve some algorithmic and computational errors. The cause of these errors are:

- Forgetting a branch condition in an If-Else statement.
- Calculating the wrong number of loop that should be taken in a For or While loop.
- Miscalculating the number of elements that an array can hold, causing overflow problem to the array.

Listed below are the steps taken in uncovering algorithmic and computational errors :

(a) For functions and subroutines that involve calculation, the calculation is performed manually and then the calculation result is compared to the actual program outcome.

(b) The output is verified if it is the desired output. If it is not, 'step-through' debugging is used to check why the outcome is not the one that is wanted.

(c) The execution of the function or subroutine is stepped through and how the value of each variable changes is examined. The values of the variables are checked to determine if they change in an expected manner or the opposite.

(d) Every branch condition in an If-Else block is tested to determine whether statements for each branch perform correctly.

(e) Various inputs are tried on an If-Else block to check whether each input can be met by a branch in the If-Else block. This is to test whether all branch conditions have been considered.

(f) For a loop structure, the loop is made to execute only once and also to execute a few times. The output is then verified.

7.2 Integration Testing

The purpose of integration testing is to test the interfaces between functions or subroutines. A function or subroutine might perform correctly after unit testing but when it is called by another function or subroutine, error might occur, most probably because of the faults in the interface of the function or subroutine.

The Face Detection System use the bottom-up approach to conduct the integration testing where functions and subroutines at the lowest level are individually tested first, and then the testing would proceed with the combination of functions and subroutines at the higher level with the lowest level ones. The testing would keep moving up the hierarchy until all the functions are tested as a whole. For instance, let's assume that Function A calls Function B and Function B calls Function C. The bottom-up approach would test C first, then test B and C together and finally test A,B and C together.

Listed below are some of the elements that are tested in integration testing :

- Compatibility between the parameters data type of a function and the values that it receives.
- Compatibility between the output data type of a function and the receiving variable of the caller function.
- Compatibility between the user interface input controls data types and the parameters data types of the functions in the control objects that the user interface interacts with. For example, to convert and display the input image as chromatic space image, the pixels' value and position must be retrieve and reset accordingly by using the GetPixel and SetPixel function before it is passed to the computation of the skin probability algorithm.

The unit testing and integration testing were conducted concurrently in the Face Detection Project as it is almost impossible to test a single function or subroutine alone because a function or subroutine would normally depend on input or output from another function and subroutine. So normally a few functions and subroutines that were related to each other were tested concurrently in order to produce the wanted test results.

7.3 System testing

The main reason for system testing is to ensure that all functional and nonfunctional requirements were fulfilled, besides uncovering any undetected bugs. The system testing is done after the unit testing and integration testing had been completed and all the initial errors had been solved. This stage is where the Face Detection System is tested as a whole.Two vital system testing were conducted for the Face Detection System, namely the function testing and performance testing.

7.3.1 Function Testing

As it names suggest, the function testing focused on functional side of the system. In simple words, it simply tests if the system had fulfilled the functional requirements correctly. The use cases specified in Chapter 3 were referred to during the function testing so that no use case would be left out from the system

7.3.2 Performance Testing

The purpose of the performance testing is to measure the accuracy of the Face Detection System in detecting face(s) as it is not 100% accurate. Besides, through these test it is found that the optimal skin threshold value is 0.25. All the test images used complies to the project scope which stated that the image used must be 24 bit RGB colour image with the width and the height of the image not exceeding 319 and 239 respectively in pixel unit.

Test to determine optimal skin threshold value:

Table 7.1 shows two of the test results to find the best skin threshold value.

Table 7.1 Test Image #1 ("lindsay.bmp")

Test	Threshold Value	% Detected face	Non-face detected
1	0.10	33	4
2	0.15	50	4
3	0.20	50	2
4	0.25	66	1
5	0.30	66	2
6	0.35	66	2
7	0.40	50	2
8	0.50	50	2
9	0.60	33	2
10	0.70	33	1
11	0.80	16	1
12	0.90	0	1
13	1.00	0	1

Table 7.2 Test Image #2 ("training5small.bmp")

Test	Threshold Value	% Detected face	Non-face detected
1	0.10	16	5
2	0.15	16	3
3	0.20	50	2
4	0.25	66	2
5	0.30	66	3
6	0.35	50	3
7	0.40	33	3
8	0.50	33	3
9	0.60	33	5
10	0.70	16	5
11	0.80	16	5
12	0.90	0	2
13	1.00	0	1

The above test was run on 6 test images and the results ere similar to that of the results shown above in the tables. Each test image is tested with 13 different threshold value and the number of percentage of detected face were calculated and number of false face detected were recorded.

From these test results, it is clear that the optimal skin threshold value for this system occurs at 0.25 because the highest percentage of faces is detected and the least amount of non-faces is found at this value.

Test to evaluate accuracy in detecting face:

After determining the optimal skin threshold value, this threshold value is used to test the accuracy of the Face Detection System. The test set is used which are:

i) Test Set A : Image which has only one face

ii) Test Set B : Image which has multiple faces

Test Set A consists of 14 test images. Test Set B consists of 20 different images with a total of 135 faces. Table 7.3 shows the test results for Test Set A and Table 7.4 shows the test results for Test Set B.

Table 7.3 Test Result for Test Set A

Image-Filename	Faces Present	Faces Correctly Detected	False Positives	False Negatives	%Detected face
abed_sBit	1	1	4	0	100
betke	1	1	3	0	100
billm	1	1	4	0	100
boy1	1	1	9	0	100
buzan	1	1	8	0	100
image1a	1	1	0	0	100
image2b	1	1	0	0	100
image4a	1	1	2	0	100
image4b	1	1	1	0	100
image9a	1	1	1	0	100
specky	1	1	1	0	100
johnm	1	1	0	0	100
man1	1	1	3	0	100
portillo	1	1	2	0	100

Table 7.4 Test Result for Test Set B

Image-Filename	Faces Present	Faces Correctly Detected	False Positives	False Negatives	%Detected Face
fatherSonplay	2	1	5	1	50
frontporch	3	3	5	0	100
Group2	8	3	3	5	38
lindsay	6	4	1	2	67
family	2	2	3	0	100
Group3	8	2	8	6	25
training2small	6	3	1	3	50
training3temp	7	5	1	2	71
cybernight1	5	5	1	0	100
cybernight2	12	10	0	2	83
cybernight3	9	9	1	0	100
training4small	18	7	2	11	39
training5small	6	4	3	2	67
training51small	7	4	5	3	57
twopeople1	2	2	5	0	100
women	2	1	4	1	50
joshua	5	4	2	1	80
dutchparty	7	5	2	2	71
graduation	19	10	0	9	53
boychair	1	1	6	0	100

As shown in Table 7.2, the system is able to successfully detect human face in all single face test images. On the other hand, Table 7.3 shows that the test yields varied results for multiple face images. The overall accuracy of the system is 66.4%. To evaluate the overall performance of the Face Detection System, the results of Test Set A and Test Set B were combined and the results are as shown below:

Total test image	34
Total faces	149
Detection Rate	66.4%
Miss Rate	33.6%

To determine if the face was detected, each face inside the final result green boxes were counted. Some images included more than one face in a box, but for this test, these faces are considered detected in this statistic. Also, some boxes only included part of the person's face. For example, some images cut off the person's chin, but included all other facial features. These images were also determined to be a "detected face" when measuring the accuracy of the program. In these images, non-face areas or better known as false positive were also detected. However, for this project it is important that the system is able to detect face with minimal miss rate (false negative which refers to face not being detected).

7.4 Discussion

The performance test which was done on the Face Detection System has shown that the system is able to find faces in a colour image with an accuracy of 68.4%. The program works well with images with only one person and a solid background. The program seems to have problems when the background has colors that are probable skin region colors.

Since the code thresholds the areas and then labels each area, neighbouring skin regions are sometimes combined. What happens in this case is that the program labels two probable skin regions as one region, which can change the shape of the area to not depict the actual object. This could lead to problem like the face combined with non-face object in the background. This changes the appearance of the segmented face shape and it does not match with the predefined oval shape, causing it not detected as face.

The detection of false face is also tested to be quite rampant. This is mainly due to objects that has similar colour as human skin and also due to regions which

were indeed skin regions but are arms and neck. Moreover when two people are very close to one another, more than one face are detected in a box.

7.5 Chapter Summary

The performance of the Face Detection System is solely dependent on its accuracy in detecting faces in an image. Therefore, the system was tested rigorously in the performance testing phase. Besides, the system had also undergone unit testing, integration testing and function testing in order to ensure the system is bug free and fulfils its functional and non-functional requirements.

CHAPTER 8

System Evaluation and Conclusion

8.1 Results

The following are examples of how the program works. The first image is the original image that is being tested for instances of a face. The second image is the grayscale skin segmented image. The lighter the area in this image, the more likely that that region is a skin region. The third image shows a binary image where skin region is segmented from the rest of the background. The skin regions are represented by white regions here. Finally, the last image is the original image with a box around the regions determined to be a face.

As shown in Figure 8.1 Result 1 below, the program worked relatively well as the image has only one face and it is the face is in upfront position. Moreover the background is controlled. The only minor set back is that the output shows one face detected in two boxes. This could be due to the constrain set in the program for the position of white and black pixels which constitutes to an oval shape. Nevertheless, as stated in Chapter 7-Testing, the face is considered detected successfully .

Original Image

Skin Likelihood Image (Graysacle Image)

Skin Segmented Image(Binary Image)

Final Detection

Figure 8.1 Result 1

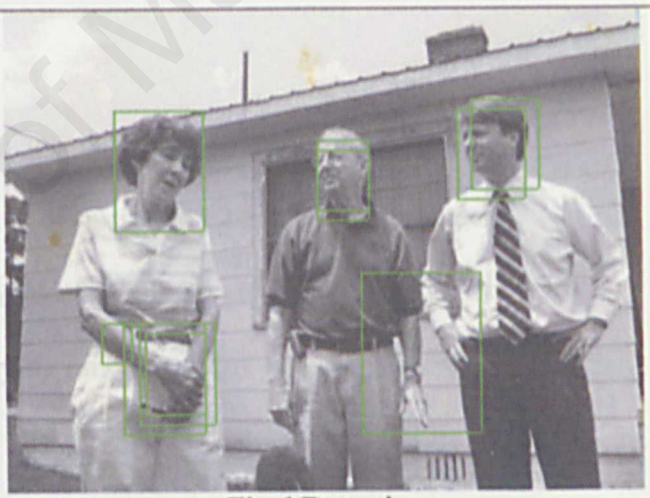
For Figure 8.2 Result 2, the program demonstrates how multiple instances of faces can be detected. This also shows that there are two very different skin color regions in the image, and both regions are detected. The only error in this image was that two persons' arms were also detected as face. The reason for this false result is because the woman's arm is crossing her pants, creating a joined skin region with some black holes in the segmented image (see segmented image). The image also has a large area detected on the two men's arms. The 2 men's arms are close to each other

causing the segmented region to have white and black pixels a positions which usually constitutes to a face.

Original Image

Skin Likelihood Image (Graysacle Image)

Skin Segmented Image(Binary Image)



Final Detection

Figure 8.2 Result 2

8.2 Strength and weaknesses of system

Strength:

The system provides a easy-to-use user interface. User do not need prior knowledge of face detection techniques in order to use the system effectively. The design are straightforward so as not to confuse users on the functions of different control on the interface. User can conveniently flip through the different image generated at different phase of the face detection (e.g: binary image, grayscale image), as each image is display on its individual tab.

Weaknesses:

The system does not achieve 100% detection accuracy. Many reasons constitutes to a less than perfect accuracy some of which were discussed in Chapter 7. Nevertheless, to date there is still no face detection system which is able to produce100% accuracy, and many researches in this field is still carrying on.

8.3 Future application

This type of program would work best for taking the first step in face recognition. For example, if a door was to be opened by a new security system, this program could be used. This program would be implemented by having the person stand in front of the camera (preferably with a solid background) next to the door and take a frontal picture of the persons face. This program would detect the positioning of the person's face in the image and then the face recognition process could begin.

8.4 Conclusion

The Face Detection System project has been successfully implemented. It is able to find instances of front facing faces in colour images with an accuracy rate of 68.4%. The Face Detection System Project has adopted the widely used technique of skin segmentation combined with the template matching technique as the method in detecting face. The system has tested to be able to perform its core function which is to detect faces in a RGB colour image. Hence, the Face Detection Project has achieved its project objectives as stated in Chapter 1.

Due to the complexity of the process, some source code were adapted from [24] to serve as guideline for programming the functions in this system. A great deal of time was spent on learning to implement the face detection process and also to understand the process of different existing face detection systems. Initially, many problems occur when trying to implement the concept into source codes as it does involve some advance syntax. Therefore, a lot of learning is done through understanding the samples and examples provided in [25].

All in all, this project has been a very interesting and valuable learning experience.

Appendix A

User Manual

This user manual provides a comprehensive guide on how to use the Face Detection System. It is presented according to the functions of the system. All the names of buttons, drop down list and other input controls on the screen would be written in quotes sign (‘ ’)

Introduction to Face Detection System

The Face Detection System is an application which detects the presence of human faces in an image file. The input image must be 24-bit RGB colour image in bitmap format. The height of the image must not exceed 239 pixel unit and the width must not be more than 319 pixel unit.

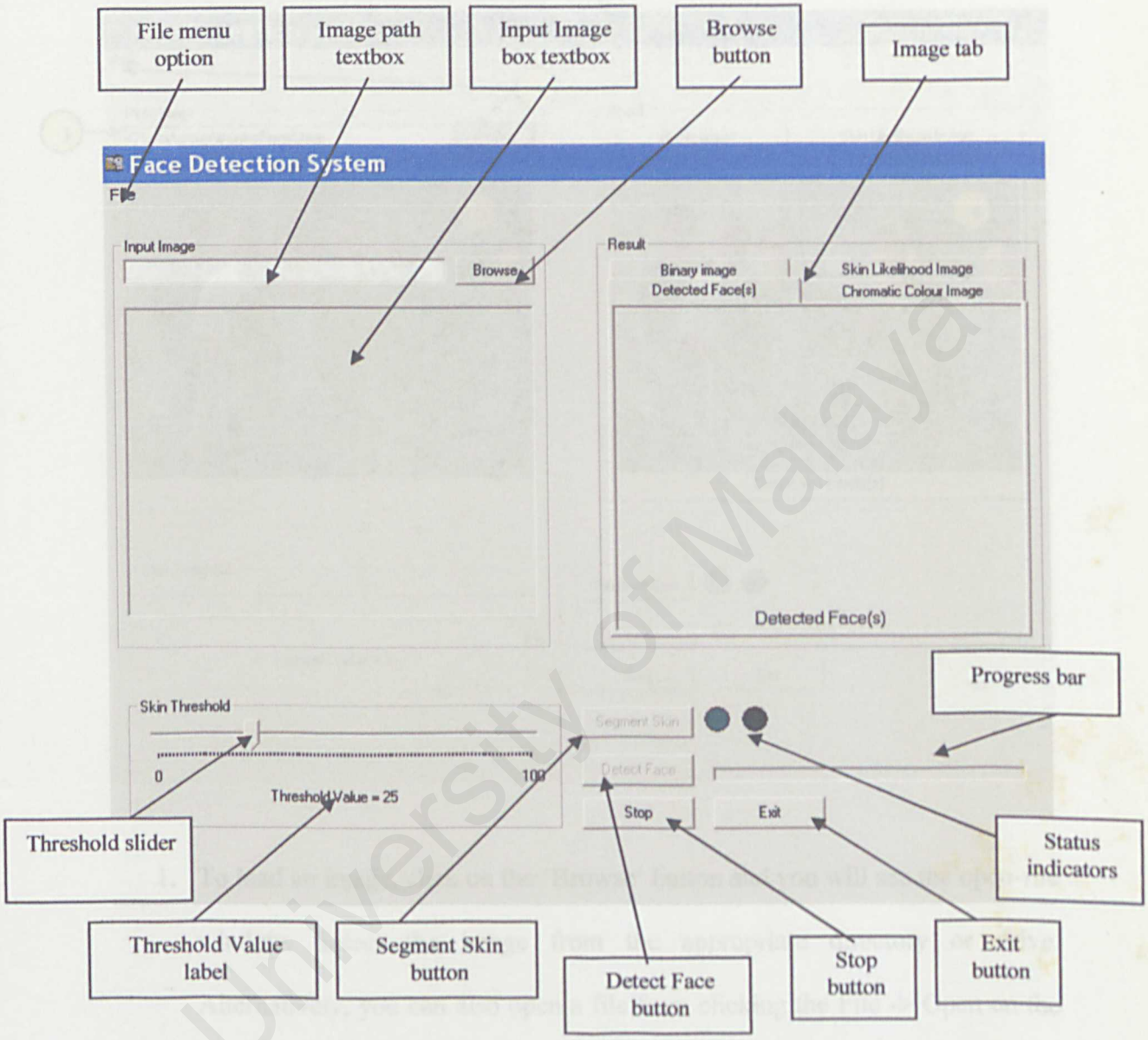
Running the Face Detection System

Insert the Face Detection System CD into your computer’s CD –ROM drive. Explore the CD and you will see the following file:

Face Detection System.exe

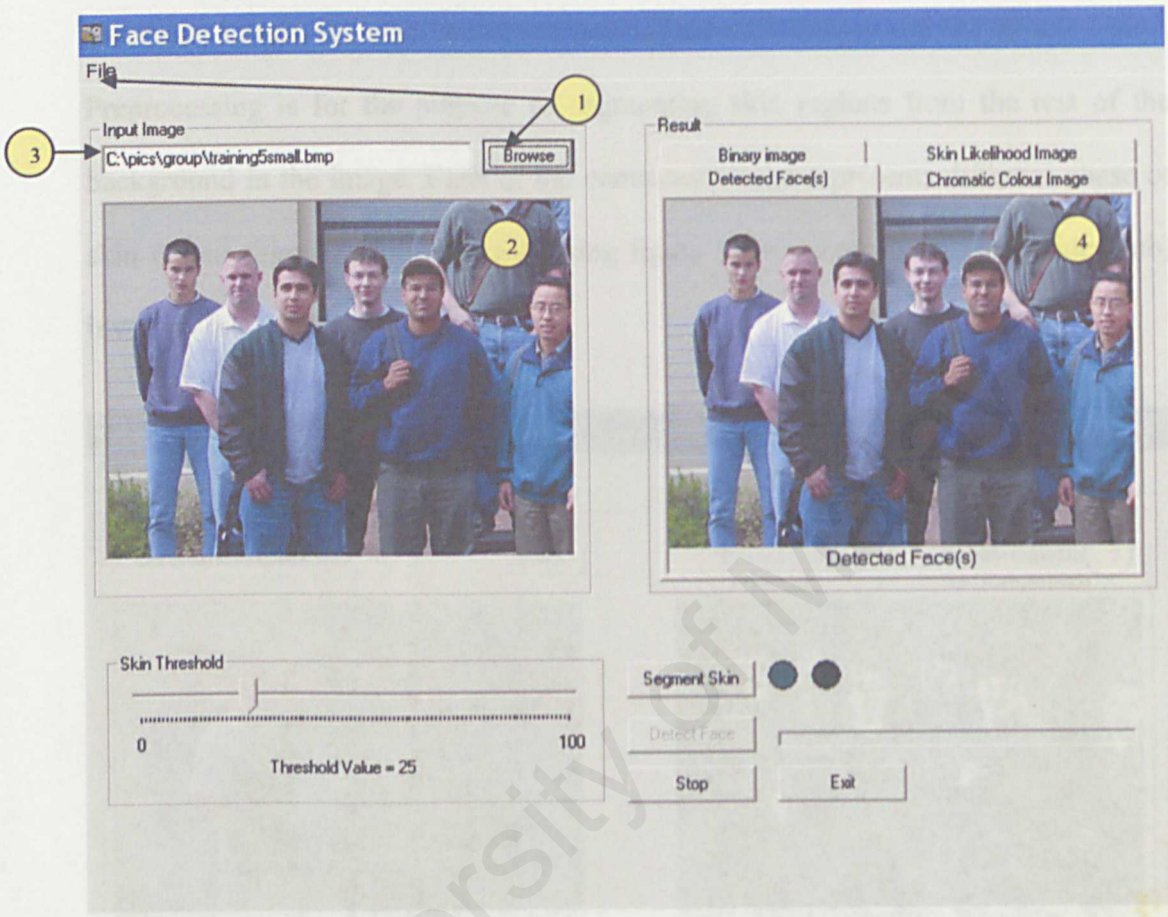
Double-click on the Face Detection System icon to run the application.

The following figure shows the different controls found on the Face Detection System.



Load Image

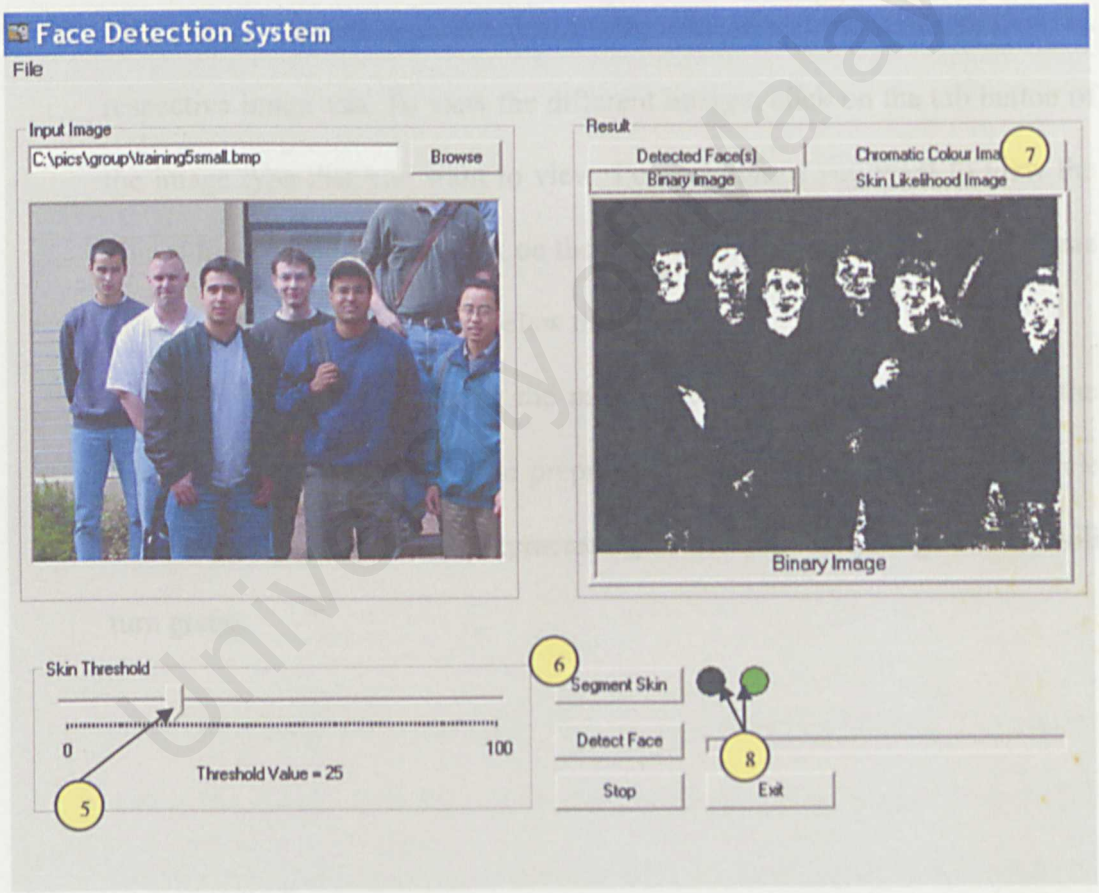
The following figure shows the interface of the Face Detection System.



1. To load an image, click on the 'Browse' button and you will see the open-file window. Select the image from the appropriate directory or drive. Alternatively, you can also open a file from clicking the File -> Open on the menu bar.
2. The loaded image will be display on the input image box
3. The path of the image file will be displayed here on the input textbox.
4. The input image will be duplicated and display on the 'Detected Face' tab to prepare it for detection.

Preprocess Image

The image that has been loaded must be preprocessed into the appropriate representation. The preprocessing converts the image into 3 different image type: the chromatic image, the skin likelihood image(grayscale image) and the binary image. Preprocessing is for the purpose of segmenting skin regions from the rest of the background in the image. Each of the converted image represents different phase of skin colour segmentation. The following figure shows controls(buttons, slider, tab) involved in the image preprocessing.

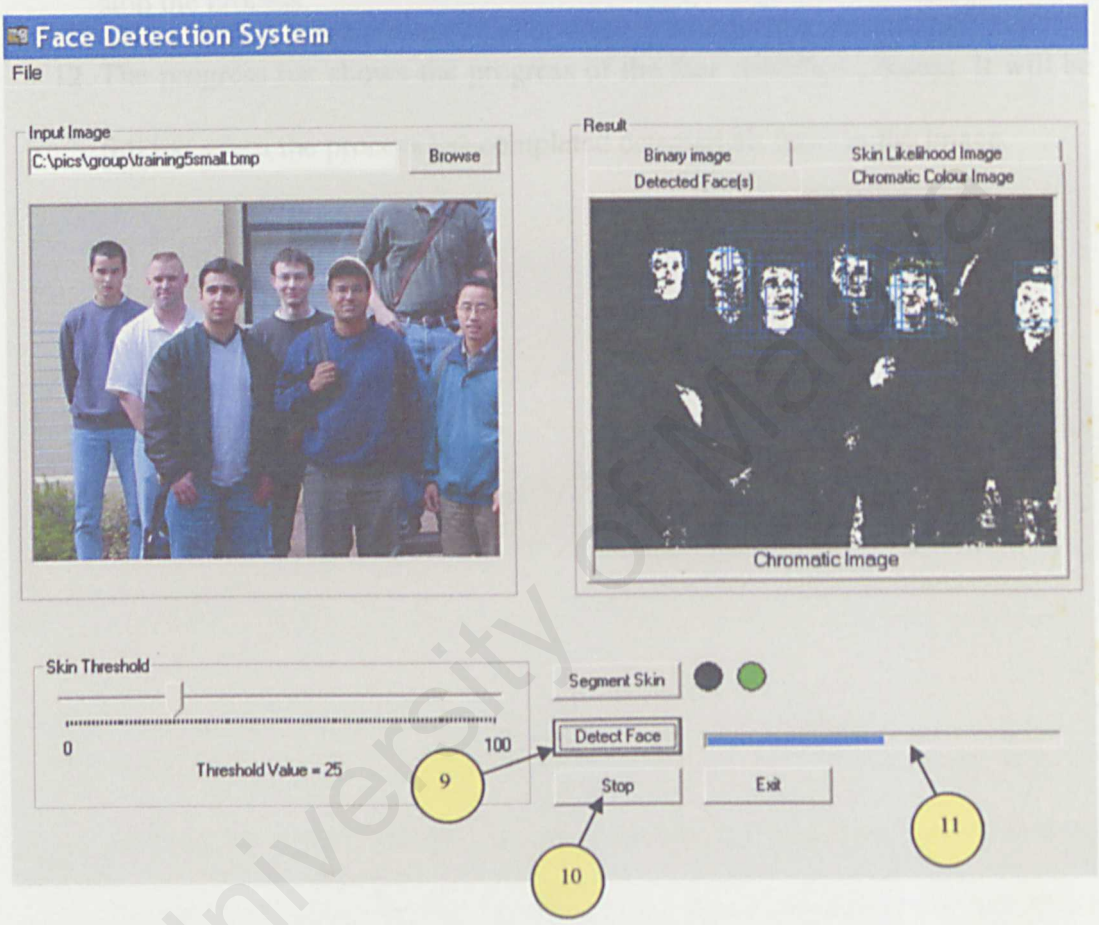


5. User can adjust the skin *threshold*[*] value. By default, the skin threshold value is set at 25 which has been tested to be the optimal value for all tested images. The value is usually adjusted for testing purposes where the user wants to find out the optimal threshold value for a particular image. The adjusted value is shown by the threshold value label under the slider. If you do not wish to adjust the threshold value, you can proceed straight to segment the image.
6. Click the "Segment Button" to preprocess the image. The left status indicator circle will turn red signifying the image is being preprocessed. Once the preprocessing is completed, the right status indicator circle will turn green.
7. The three different representation of the image will be displayed on its respective image tab. To view the different images, click on the tab button of the image type that you want to view. For example if you want to view the binary form of the image, click on the "Binary Image" tab button . The name of each image will be labeled below the image.
8. The status indicator circle, as the name implies, indicate the status of the image preprocessing. When the preprocessing is still running, the left circle will remain red. Once the preprocessing is completed, the right circle will turn green.

[*] *Threshold is the parameter of brightness for an image. Different images require different threshold. If the threshold value is too low, the amount of segmented skin regions increases and if it is too high certain skin region will not be segmented.*

Detect Face

The system can only detect face after the image has been preprocessed. Therefore, before and throughout the image preprocessing, the "Detect Face" button is disabled. Once the preprocessing is completed, the "Detect Face" button will be enabled.

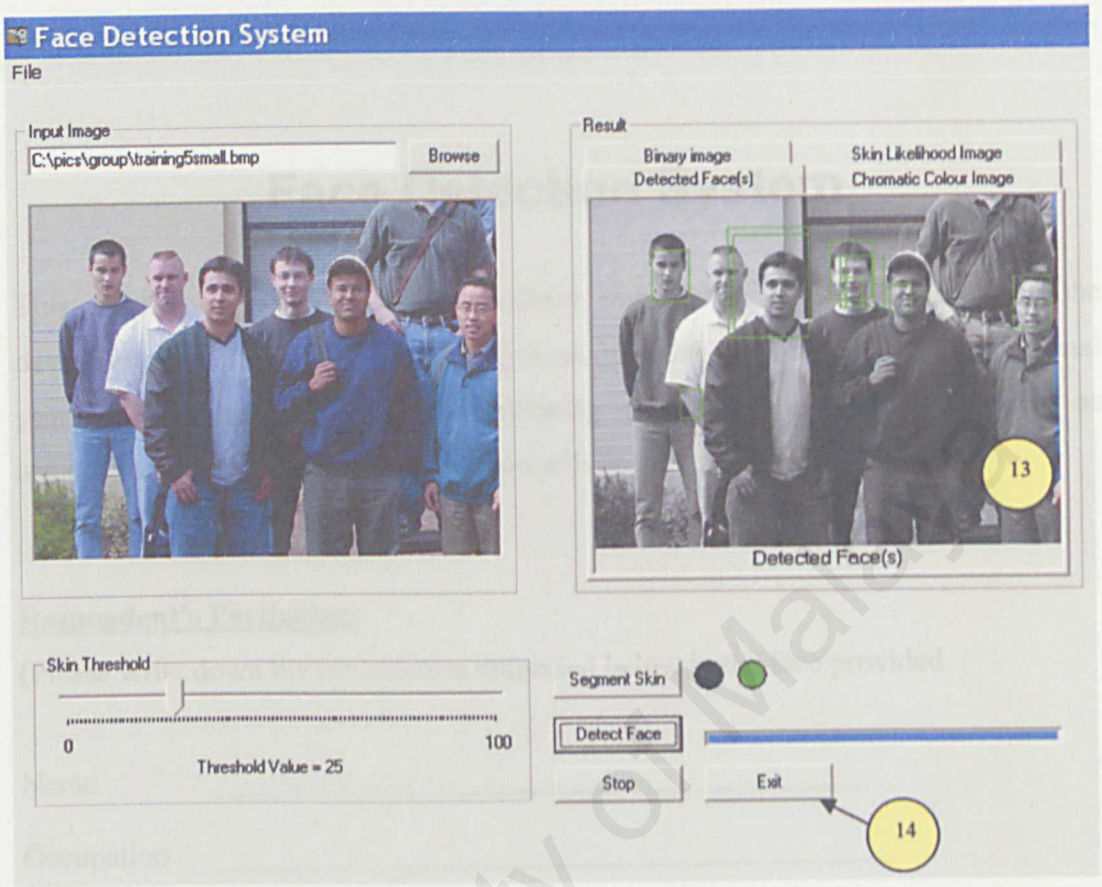


9. Click the "Detect Face" button to start the face detection process. The system will automatically scan through the binary image to find possible faces. The process will take some time (approximately 10 to 40 seconds-depending on the image).
10. The "Chromatic Image" tab will show the system scanning the image to search for faces as the detection process is running. The "Cromatic Image" in the figure above is an example of the system scanning the image. A box is

drawn around every potential face. The colour of the box represents the degree probability of the "boxed region" to be a face. Green box represents the highest probability, blue represents medium probability and light blue represents lowest probability.

11. As the face detection process is running, user can click the "Stop" button to stop the process.
12. The progress bar shows the progress of the face detection process. It will be full bar when the process has completed detected all faces in the image.

Final Output



13. Once the detection process is completed, the system will output the detected faces in the "Detected Face" tab. A green box is drawn around every face hat the system is able to detect. As shown in the figure above, it is the final result.
14. To exit the application, click on the "Exit" button. Alternatively, user can also quit the application by clicking on the File -> Exit on the menu bar. If user wants to continue using the application to detect face in other image, user just need to load a new image.

Appendix B

Face Detection System

This questionnaire is designed to obtain some input from the public on the development of the Face Detection System. It is the proposed system for my final year project. Please complete the following questions to reflect your opinion as accurately as possible. Your information will be kept strictly confidential.

Respondent's Particulars

(Please write down the information requested below in the face provided.

Name _____

Occupation _____

1. Please tick the answer that fits your opinion

	Yes	No
a) Do you access the internet?	☐	☐
b) Do you own a computer at home?	☐	☐
c) Do you know what is image processing?	☐	☐
d) Have you come across a face processing system before? (i.e: face recognition, face detection, etc.)	☐	☐
e) Would you prefer using a system which is dialog driven?	☐	☐
f) If given the opportunity, would you be interested in	☐	☐

learning more about face detection?

2. Will you choose face detection as your final year project or research topic? Why

_____

_____

3. Are you aware of the technique to detect face? (please tick your choice)

Yes____ No____

If 'yes' proceed to question 4. If 'no', go to question 5

4. Which face detection technique which you know of?

____ Template matching

____ Skin colour analysis

____ Neural networks

____ Eigenface approach

____ Support Vector Machine

____ Others. Please specify_____

5. What functionality would you like to have on a face detection system?

_____

_____

6. Depending on which technique is used, a face detection process will usually generate different types of image (i.e: binary image, gray-scale image, etc) from the original input image.

Would you want to have the options to view them?

____Yes *(please proceed to question 8)* ____No *(please go to question 9)*

7. How would you prefer to view want to view the images?

____ on a dialog box

____ on a separate window

____ others.Please specify

8. What type of input image would you prefer?

____Bitmap

____ Colour RGB

____ JPEG

____ Others. Please specify

9. Would you want to have a status bar indicating that the system is running

____Yes ____No

10. Would you want a help file to be included in the system?

____ Yes ____No

The End

~ Thank you for your participation ~

REFERENCE

[1] Ming-Hsuan Yang, David Kriegman, and Narendra Ahuja, "Detecting Faces in Images: A Survey", *IEEE Trans. Pattern Anal. and Machine Intell.(PAMI)*, vol. 24, no. 1, pp. 34-58, 2002.

[2] I.A. Essa and A. Pentland, "Facial Expression Recognition Using a Dynamic Model and Motion Energy," *Proc. Fifth IEEE Int'l Conf. Computer Vision*, pp. 360-367, 1995.

[3] H.A. Rowley, S. Baluja and T. Kanade, "Neural Network Based Face Detection," *IEEE Trans. Pattern Anal. and Machine Intell.(PAMI)*, 1998

[4] Edgar Osuno, Robert Freund and Fredico Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proceedings of CVPR'97*, June 17-19,1997.

[5] Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process,* Prentice Hall, Upper Saddle River, NJ, 2002, pp. 18-22

[6] D.Saxe and R Foulds, "Towards Robust Skin Identification in Video Images," *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pp. 379-384, 1996.

[7] Jamie Sherrah and Shaogang Gong, "Skin Colour Analysis," http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/GONG1/cvOnline-skinColourAnalysis.html

[8] M. Turk and A. Pentland, "Eigenfaces for Recognition," J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

[9] http://www-white.media.mit.edu/vismod/demos/facerec/basic.html

[10] Washington Technology http://www.washingtontechnology.com/news

[11] http://www.stanford.edu/class/ee368/Project/slides/ee368group07.ppt

[12] http://egweb.mines.edu/eges512/projects/face/cobb.pdf

[13] http://www-cs-students.stanford.edu/~robles/ee368/main.html

[14] http://computer.howstuffworks.com/facial-recognition.htm

[15] Si Alhir, Sinan. (2002). Understanding the Unified Process. *Methods and Tools*. **10(1)**. 2-17

[16] Ivar Jacobson, "Applying UML in the Unified Process", http://www.compapp.dcu.ie/~renaat/gdip/UMLconf.ppt

[17] Jacobson, Ivar., Booch, Grady. & Rumbaugh, James. (1999). *The Unified Software Development Process*. Addison Wesley

[18] Object Oriented Analysis and Design team, "What is UML?" http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/ (Current Spring 2001)

[19] Leffingwell, D.,Widrig, D.(2000),*Managing software requirements -A Unified Approach,* Upper Saddle River,NJ, Addison-Wesley

[20] G. Wyszecki and W.S. Styles. Color Science: Concepts and Methods, Quantitative Data and Formulae, second edition, John Wiley & Sons, New York 1982

[21] Azwina Mohd Yusof, "Face Recognition and Identification Using Eigenfaces", Dissertation, University of London, 2001

[22] R. Ramesh, Kasturi R. and Schunck B., Machine Vision, pp 31 - 51, McGraw

[23] Hill, New York 1995

[24] http://www.fuzzgun.btinternet.co.uk/

[25] http://www.planet-source-code.com/vb

[26] http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip.html

BIBLIOGRAPHY

Umbargh, S.E., *Computer Vision and Image Processing*, Prentice Hall, 1998.

Taylor, Dena(2001). The Literature Review: A Few Tips On Conducting It. [Online], Available: http://www.utoranto.ca/litrev.html

Ablan, J., Xie,D., Payet, R., et al., *Professional Visual Basic 6: The 2003 Programmer's Resource,* Wrox Press Ltd, 2003.